Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

DSSP50 Build 090910 Documentation

Part 1: Guide and Tutorial

M. Mackay

Defence R&D Canada – Atlantic

Technical Memorandum
DRDC Atlantic TM 2009-226
December 2009

Canada

This page intentionally left blank.

DSSP50 Build 090910 Documentation, Part 1: Guide and Tutorial

M. Mackay
Defence R&D Canada – Atlantic

Defence R&D Canada – Atlantic
Technical Memorandum
DRDC Atlantic TM 2009–226
December 2009

Author

Original signed by M. Mackay

M. Mackay
Defence Scientist

Approved by

Original signed by James L. Colwell for

N. Pegg
Head, Warship Performance

Approved for release by

Original signed by Ron Kuwahara for

C. Hyatt
Chair, Document Review Committee

© Her Majesty the Queen as represented by the Minister of National Defence, 2009
© Sa majesté la reine, représentée par le ministre de la défense nationale, 2009

Abstract

DSSP50 is a computer code for evaluating the dynamics and manoeuverability of submarines and smaller streamlined underwater vehicles. This document is the guide to the program. It comprises an overview of DSSP50 and a tutorial. Other documents detail the input requirements and describe the underlying algorithms.

Résumé

DSSP50 est un code machine permettant d'évaluer la dynamique et la manœuvrabilité des sous-marins et des plus petits véhicules sous-marins hydrodynamiques. Le présent document est un guide pour le logiciel. Il comporte un survol du DSSP50 ainsi qu'un tutoriel. D'autres documents donnent les détails de ses exigences d'entrée et décrivent les algorithmes sous-jacents.

This page intentionally left blank.

Executive Summary

DSSP50 Build 090910 Documentation, Part 1: Guide and Tutorial
M. Mackay; DRDC Atlantic TM 2009–226; Defence R&D Canada – Atlantic; December 2009.

Introduction: DRDC Submarine Simulation Program, version 5.0 (DSSP50), was developed to predict the manoeuverability of submarines and smaller underwater vehicles. It combines the capability of two earlier codes, DSSP21 and DSSP40. The program is used to calculate loads on captive models, derive hydrodynamic coefficients, simulate tactical manoeuvers, and generate Manoeuvering Limitation Diagram (MLD) curves.

This document is the program guide to DSSP50. In addition to providing basic information on the program, it includes a tutorial that covers hydrodynamic modeling of the vehicle and running simulations.

Significance: The target user group for the program is DRDC and other Government researchers, CF engineers, and CF contractors.

DSSP50 was formulated to balance accuracy, which is sufficient for predicting complex tactical manoeuvers, with computational speed. Speed is required for systematic studies in support of preliminary design and for MLD calculations. Making it multifunctional ensured that all calculations and simulations are based on a common set of algorithms, and thus consistent.

Although input to the program initially appears complex because of a large number of options, experience with its predecessors suggests that it should be relatively easy to learn. New users are advised to do the tutorial exercises.

Future Development: Future development may include such features as near-bottom operation.

Sommaire

DSSP50 Build 090910 Documentation, Part 1: Guide and Tutorial
M. Mackay; DRDC Atlantic TM 2009–226; R & D pour la défense Canada – Atlantique; decembre 2009.

Introduction: Le logiciel de simulation sous-marine de RDDC, version 5.0 (DSSP50), a été développé afin de prévoir la manœuvrabilité des sous-marins et des véhicules sous-marins plus petits. Il combine la capacité de deux codes antérieurs, DSSP21 et DSSP40. Le logiciel sert au calcul des charges sur des maquettes entravées, à calculer les coefficients hydrodynamiques, à simuler les manœuvres tactiques et à générer des courbes des schémas de limitation des manœuvres (SLM).

Le présent document est le guide logiciel au DSSP50. En plus de fournir l'information de base sur le logiciel, il comprend un tutoriel qui couvre la modélisation hydrodynamique du véhicule et des simulations du fonctionnement.

Portée: Le groupe utilisateur cible pour le logiciel est RDDC et d'autres chercheurs du gouvernement, les ingénieurs des FC et les entrepreneurs des FC.

Le DSSP50 a été formulé pour équilibrer la précision, qui est suffisante pour prévoir des manœuvres tactiques complexes, à la vitesse computationnelle. La vitesse est requise pour des études systématiques de soutien à la conception préliminaire et aux calculs des SLM. Le fait de le rendre multifonctionnel a permis de garantir que tous les calculs et les simulations sont basés sur un ensemble commun d'algorithmes et sont ainsi compatibles.

Quoique l'entrée dans le logiciel semble initialement complexe compte tenu du grand nombre d'options, l'expérience avec ses prédécesseurs donne à penser qu'elle devrait être relativement facile à apprendre. On conseille aux nouveaux utilisateurs de faire les exercices du tutoriel.

Recherches futures: Les recherches futures pourraient comprendre des caractéristiques comme les opérations près du fond.

Table of Contents

Abstract i
Résumé i
Executive Summary iii
Sommaire iv
Table of Contents v
List of Figures vi
List of Tables vi
1 Introduction 1
2 Input and Output Files 2
3 Running DSSP50 Directly 4
4 Running DSSP50 from Another Program 4
5 Concluding Remarks 5
References 6
Annex A. Tutorial 7
Exercise 1: Component Model – Hull 11
Exercise 2: Component Model – Sail 15
Exercise 3: Component Model – Rudders 15
Exercise 4: Component Model – Sternplanes and Bowplanes 17
Exercise 5: Component Model – a Small Lump 19
Exercise 6: Component Model – X-rudders and Sailplanes 21
Exercise 7: Coefficient Model – the Whole Vehicle 24
Exercise 8: Hybrid Model – Modified Appendages 27
Exercise 9: Propulsion, Autopilots, MBTs, and other Systems 31
Exercise 10: Free Manoeuvering Simulations 35
Exercise 11: MLD Curve Calculations 47
Annex B. Program Unit Lists 55
Annex C. Error Messages 61
Nomenclature 65
Distribution 67

List of Figures

Figure A.1. Generic Submarine: elevation and plan 7
Figure A.2. Sail dimensions 9
Figure A.3. Rudder dimensions 9
Figure A.4. Sternplane dimensions 10
Figure A.5. Bowplane dimensions 10
Figure A.6. Nondimensional loads for the hull alone in pitch 14
Figure A.7. Hull + sail + rudder + sternplane geometry 17
Figure A.8. Completed geometry 20
Figure A.9. X-rudder version geometry with the mast omitted 23
Figure A.10. Hybrid model with resized sternplanes 29
Figure A.11. Hybrid model with resized sail 31
Figure A.12. Example free manoeuver time histories 40
Figure A.13. Schematic Manoeuvering Limitation Diagram 48
Figure A.14. Example flood avoidance boundary 50
Figure A.15. Example avoidance boundaries for a 10 degree jam to rise ... 53
Figure C.1. Run time error message from the Salford compiler library 63

List of Tables

Table A.1. Generic submarine hull geometry 8
Table A.2. Exercise 10 command list 39

1 Introduction

Computer program DSSP50 evaluates the hydrodynamics and manoeuverability of underwater vehicles. It is primarily for submarines although it can be applied to smaller streamlined submersibles. Hydrodynamic loads on the vehicle are calculated with a mix of fast semiempirical and coefficient based methods. Speed is necessary for systematic parameter variation during preliminary design, for evaluating emergency recovery procedures, and for other applications requiring many simulation runs.

DSSP50 documentation is in three parts:

- A program guide and tutorial, this memorandum, primarily for new users.
- An input reference, reference [1], for all users.
- A description of the algorithms, reference [2], for background information.

After some familiarity with the program is acquired, the principal resource will be the input reference.

This guide comprises an outline of the input and output files, notes on running the program, a tutorial (annex A), a list of program units (annex B), and discussion of error messages (annex C). Although the tutorial is designed as a series of hands-on exercises, it may also be useful as a review of the capabilities of the program.

The normal release of DSSP50 comprises a Windows executable `dssp50.exe` with the run-time library `salflibc.dll` and some other files including those created by the author for the tutorial.

Developmental status of the code is indicated by a build number consisting of the release date in *yymmdd* format. (In output from the program, an additional character may be appended: `A`, `B`, or `X`, indicating that the build is considered to be a preliminary, limited, or experimental release respectively. After sufficient validation the qualifier will be dropped.) The build at time of writing is 090910; subsequent builds will be accompanied by updates and additions to the documentation as required.

The program proceeds in two phases:

- Vehicle model and systems setup. The user must provide a file defining the vehicle geometry or hydrodynamic coefficients, or a combination of the two, together with basic systems information. The data is extensively checked and as much preprocessing of it is done as possible.
- Hydrodynamic and manoeuvering simulations. The simulations to be run are defined in sequence on an optional input file. Simulations include static and dynamic loads on a captive vehicle, hydrodynamic coefficient generation, free vehicle manoeuvering, and Manoeuvering Limitation Diagram (MLD) curve calculation. If the simulation input file is absent, there is no work done in the second phase.

The objective in making the program multifunctional was to base all calculations and simulations on a common set of hydrodynamic modeling subroutines. The underlying hydrodynamic models are therefore consistent, and improvements and enhancements to the models will be applied universally.

DSSP50 is programmed in Fortran. Some parts of it are from much earlier programs, other parts are quite new. The code was revised to compile under Fortran 95 syntax rules, but only a handful of features are unlikely to compile under Fortran 77. One feature, the library routine required to extract the optional command line parameter, is compiler-dependent. DSSP50 was developed on a Microsoft Windows PC using the Salford FTN95 Win32 version 2.00 compiler, and checked with STI's Understand for Fortran, version 1.4. Although the code is virtually platform-independent, the documentation assumes a Windows platform where relevant.

The following conventions in the documentation should be noted. Object names (program units, etc.) and DSSP50 keywords are in a capitalized typewriter font. In discussing input, higher level keywords [1] following a lower level one are indicated by an arrow: `Lower` → `Higher`. The words 'simulation' and 'run' are generally synonymous except for an MLD simulation which comprises a number of individual runs.

2 Input and Output Files

The naming scheme for most input and output files has the form *Root.Ext*. *Root* is a user-defined string up to eight characters long ('`dssp50`' is used by default) and *Ext* indicates the function of the file. In addition, each simulation may generate files `sim`*nnn*`.txt` and `aux`*nnn*`.txt`, where *nnn* is the number of the simulation in the input file sequence. An arbitrarily named time history input file is required for a captive simulation. Lump data can be output to or read from arbitrarily named files.

Root may be input as a command line parameter so that the program can be run from another without any user intervention. Without a command line parameter, *Root* is requested from the user. In either case, *Root* is truncated to eight characters if it is longer.

File names are interpreted as follows:

File	Function
Root`.geo`	input file for vehicle and systems setup
Root`.gpr`	vehicle and systems printer output
Root`.acd`	geometry output for `AcroSpin` plotting
Root`.gpl`	geometry output for `Maple` or `MatLab` plotting
Root`.wrl`	geometry output for `VRML` plotting

Root.sim	input file for simulation sequence
Root.spr	simulation sequence printer output
sim*nnn*.txt	tabulated output from simulation *nnn*
aux*nnn*.txt	tabulated auxiliary output from simulation *nnn*
Root.log	run log output
DSSP50.myp	a few global inputs can be included here

The absence of *Root*.geo generates a fatal error. All files except binary lump output and input consist of ASCII characters, and can be read and manipulated with a text editor. The tabular output files sim*nnn*.txt and aux*nnn*.txt can be opened directly by most spreadsheet and graphical analysis software.

For the most part, output from DSSP50 is self-explanatory. It is also too voluminous to illustrate here. For error checking, the *Root*.log file echoes all input from *Root*.geo and *Root*.sim, reports a number of intermediate calculation steps, and outputs all error messages. Most errors in input can be identified by comparing the error message and the last echoed line of input. Those in a free manoeuvering command, however, will not be flagged until the command is executed.

While a *Root*.geo file is being processed, input data, intermediate results, and hydrodynamic and other characteristics are output to *Root*.gpr. The vehicle components, including lumps, and systems therefore appear in the same order as on the input file. At the end, there is a summary of propulsor initialization, a table of overall vehicle properties, tables of the added and inertial masses and moments of inertia, if calculated, and a table of autopilot parameters. The level of detail in *Root*.gpr has been useful while the code is under development, but is probably too much for most purposes. Some of it may be made optional in later builds.

A summary of conditions and results for each of the runs called for in *Root*.sim is given on *Root*.spr. Its purpose is to confirm that the run parameters and other conditions were properly set up, especially if there are anomalies in the results. The principal output of calculations and simulations is to the sim*nnn*.txt and aux*nnn*.txt files, and is in simple tabular form for importing into spreadsheet, graphical, or other post-processing software.

Root.spr starts with a vehicle configuration summary that includes a list of components by number and label, indirect deflection control weights, a table of appendages associated with various quasi-direct and indirect commands, and reference axes information. For each simulation, the number *nnn* of the corresponding sim*nnn*.txt and aux*nnn*.txt files is given. Static output merely lists values of the simulation parameters and the number of iterations if a Loop is executed. Coefficnt output lists all the coefficients and stability indices calculated. Free simulation output includes initialization data and the target trajectory time history in chart axes, i.e., (t, x_T, y_T, z_T), at each time, or interpolation, step. MLD simulation output tabulates the MLD curve.

A `simnnn.txt` file tabulates loads for a static run, coefficients (if the `CoefOut` option is selected [1]) for a coefficient run, and load time histories for a captive run. For a free simulation it tabulates time histories of the extended state variable data: time, speed, (x_0, y_0, z_0) coordinates, Euler angles, compass heading, translational and rotational speeds in boat axes, control deflections, and RPM. An `auxnnn.txt` file tabulates time histories of some useful auxiliary data: chart coordinates, power, thrust, autopilot error and control signals, bottle group pressures and temperatures, water displaced from ballast tanks and floodwater mass. The plot-ready MLD curve data is saved on `simnnn.txt`, and either a MLD run summary or run time histories are saved on `auxnnn.txt`.

In a free or MLD simulation, some data in the `auxnnn.txt` file may be approximated during interpolation. Although the program recalculates those quantities for which the approximation would lead to an unacceptable loss of precision, it is not done for those that are considered to be primarily for qualitative or diagnostic purposes, e.g., thrust, power, autopilot signals, or changes in vehicle mass.

3 Running DSSP50 Directly

DSSP50 has been written to run from the input files with virtually no user intervention. Unless the *Root* character string that is required to identify input and output files is detected as a command line parameter, the user is asked to supply it at the start of the program:

```
Enter file name root (default = dssp50):
```

A string up to eight characters long is expected. Enter/Return indicates the default '`dssp50`'. Not finding the geometry input file *Root*`.geo` generates a fatal error.

There are a few conditions that generate a warning followed by a pause, at which point the user is required to intervene. A typical example is:

```
Calculate residual drag for truncated hull ... (Y/[N])?
(It may be incorrect)
```

The program can be set to automatically take the default action for all these by adding `Warnings Quiet` to the input files or to `DSSP50.myp`.

4 Running DSSP50 from Another Program

DSSP50 was developed on, and to date has only been tested on, a Microsoft Windows platform. The command line parameter option discussed in this section is compiler and operating system dependent; if the program is migrated to a another platform a substitute may function differently.

The *Root* character string that is required to identify input and output files may be supplied as a command line parameter, allowing the program to be run without user intervention unless an error occurs. In the following examples, *Root* is the string '`demo`', so that a file `Demo.geo` must be supplied to define the vehicle, and another, `Demo.sim`, will, if found, be used to specify the simulations. (Filenames are case insensitive.)

Within MATLAB (`http://www.mathworks.com`) — assuming that the `DSSP50` executable and input files are in the MATLAB current directory — the program can be called with the required *Root* from the command window using the exclamation point operator:

```
! dssp50 demo
```

The same sequence works in a MATLAB M-file. Other programs can call `DSSP50` in a similar fashion.

The command line parameter is also useful for running `DSSP50` standalone. If many runs are to be made with the same *Root*, presumably editing the input files between runs, simply double-clicking the following batch file in Windows Explorer will save having to enter *Root* each time.

```
@echo off
if exist demo.geo dssp50 demo
:END
```

The first and last lines are not really needed. Testing for the existence of `Demo.geo` is also optional since the program will stop with an error if the vehicle definition file is not found. Input files must be in the same directory as the batch file. If the `DSSP50` executable is not, it must be on the default path or have its path specified in the batch file.

5 Concluding Remarks

This is the first of three documents describing build 090910 of the underwater vehicle manoeuvering simulation code `DSSP50`. It is the program guide, which is primarily intended to give new users an overview of the program together with some hand-on experience by means of the tutorial contained in annex A.

There are still some capabilities, such as near-bottom operation, that can be added to the program. Others may undergo improvement or additional validation. Nevertheless, the current build is sufficiently complete to be benchmarked at this point before further development.

References

1. Mackay, M. (2009). DSSP50 Build 090910 Documentation, Part 2: Input Reference. (DRDC Atlantic TM 2009-227). Defence R&D Canada – Atlantic.

2. Mackay, M. (2009). DSSP50 Build 090910 Documentation, Part 3: Algorithm Description. (DRDC Atlantic TM 2009-228). Defence R&D Canada – Atlantic.

3. Mackay, M. (2003). The Standard Submarine Model: A Survey of Static Hydrodynamic Experiments and Semiempirical Predictions. (DRDC Atlantic TR 2003-079). Defence R&D Canada – Atlantic.

4. Mackay, M. (2006). DSSP21 Build 061102 User Guide. DRDC TM 2006–252. Defence R&D Canada – Atlantic.

5. Mackay, M. (2001). Some Effects of Tailplane Efficiency on Submarine Stability and Manoeuvering. (DRDC Atlantic TM 2001-031). Defence R&D Canada – Atlantic.

6. Hoerner, S.F. (1958). *Fluid-Dynamic Drag.* Midland Park, NJ: published by the author.

7. Mackay, M., Williams, C.D., and Derradji-Aouat, A. (2007). Recent Model Submarine Experiments with the MDTF. In *Proceedings of the 2007 Canadian Marine Hydrodynamics and Structures Conference.* St. John's, NL.

8. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G. (1986). Interpolants for Runge-Kutta Formulas. *ACM Transactions on Mathematical Software*, Vol. 12, No. 3. Association for Computing Machinery.

9. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G. (1988). Effective Solution of Discontinuous IVPs Using a Runge-Kutta Formula Pair with Interpolants. *Applied Mathematics and Computation*, Vol. 27. Amsterdam: Elsevier.

10. Watt, G.D. (1988). Estimates for the Added Mass of a Multi-Component, Deeply Submerged Vehicle, Part I: Theory and program Description. (DREA TM 88/213). Defence Research Establishment Atlantic.

Annex A. Tutorial

To keep this tutorial to a reasonable length it is assumed that the user has at least the input reference [1] at hand. It will be useful to also have the algorithm description [2]. With this in mind, most of the input is presented without repeating the information in these references. Furthermore, only commonly used input options are cited — the user is encouraged to try others in subsidiary exercises — and discussion of output from the program is limited to key points.

It is necessary to have a reasonably capable text editor such as Context or Crimson Editor (`www.contexteditor.org`, `www.crimsoneditor.com`). There are many others freely available. The user should associate input and output file extensions with the editor. A text file comparator such as ExamDiff (`www.prestosoft.com`) will be useful to quickly identify changes in input or output.

Different formats for plotting geometry (of a component model) are discussed in reference [1]. The examples here use VRML output rendered by the Myriad 3D Viewer (`www.myriadviewer.com`). Tabulated results can be read by a spreadsheet for plotting and analysis, but the author has found specialized data manipulation software such as Origin (`www.originlab.com`) to be more effective.

We start by building a component model of the generic submarine illustrated in figures A.1–A.5. Users who are only interested in a coefficient model can go directly to exercise 7, but should note that some useful general concepts are developed in the intervening exercises.

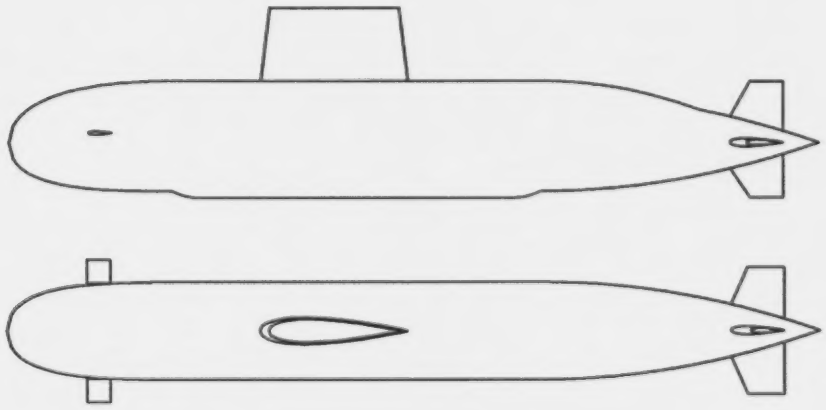


Figure A.1. Generic Submarine: elevation and plan.

Table A.1. Generic submarine hull geometry: breadth (B), upper and lower offsets, height (T), area (A), and camber (F), *vs* x. Dimensions are in m, area in m^2.

x	breadth	upper	lower	height	area	camber
0.00	0	0	0	0	0	0
−2.03	5.1640	−2.8864	2.5820	5.4684	22.1788	−0.1522
−4.06	6.3437	−3.7256	3.1719	6.8975	34.3658	−0.2769
−6.10	6.9018	−4.1988	3.4509	7.6497	41.4665	−0.3740
−8.13	7.1675	−4.4708	3.5838	8.0546	45.3421	−0.4435
−10.16	7.2753	−4.6087	3.6377	8.2464	47.1201	−0.4855
−12.20	7.3000	−4.6500	3.6500	8.3000	47.5872	−0.5000
−14.23	7.3000	−4.6500	4.1500	8.8000	50.4538	−0.2500
−16.26	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−18.30	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−20.33	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−22.36	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−24.40	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−26.43	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−28.46	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−30.50	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−32.53	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−34.56	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−36.60	7.3000	−4.6500	4.1500	8.8000	50.4540	−0.2500
−38.63	7.3000	−4.6500	4.0999	8.7499	50.1665	−0.2751
−40.66	7.2944	−4.6441	3.6472	8.2913	47.5006	−0.4985
−42.70	7.1859	−4.5423	3.5930	8.1353	45.9143	−0.4747
−44.73	6.9395	−4.3142	3.4698	7.7839	42.4245	−0.4222
−46.76	6.5551	−3.9596	3.2775	7.2371	37.2591	−0.3410
−48.80	6.0326	−3.4785	3.0163	6.4949	30.7728	−0.2311
−50.83	5.3722	−2.8711	2.6861	5.5572	23.4476	−0.0925
−52.86	4.5738	−2.2869	2.2869	4.5738	16.4300	0
−54.90	3.6373	−1.8187	1.8187	3.6373	10.3909	0
−56.93	2.5629	−1.2814	1.2814	2.5629	5.1588	0
−58.96	1.3504	−0.6752	0.6752	1.3504	1.4323	0
−61.00	0	0	0	0	0	0

This hull is the axisymmetric standard model hull [3] scaled to 61 m length and 7.3 m breadth, with a 1 m deep casing and a 0.5 m deep keel added.

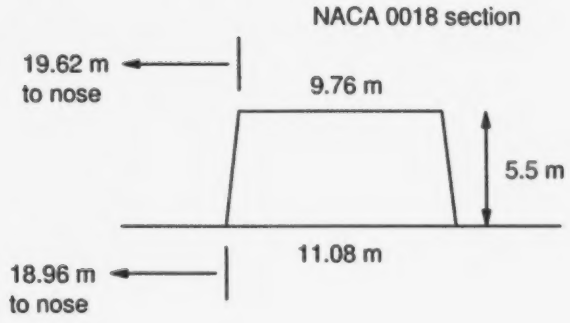


Figure A.2. Sail dimensions.

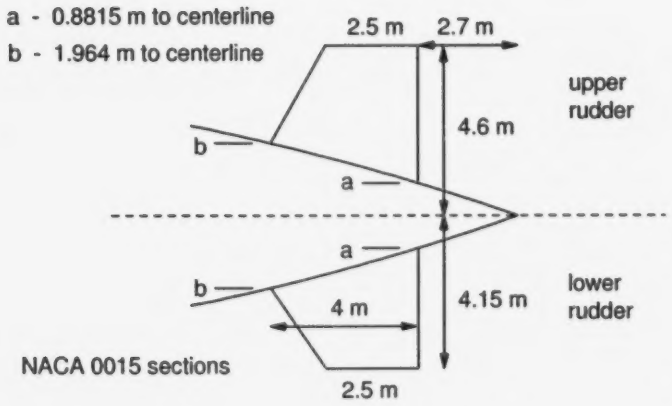


Figure A.3. Rudder dimensions.

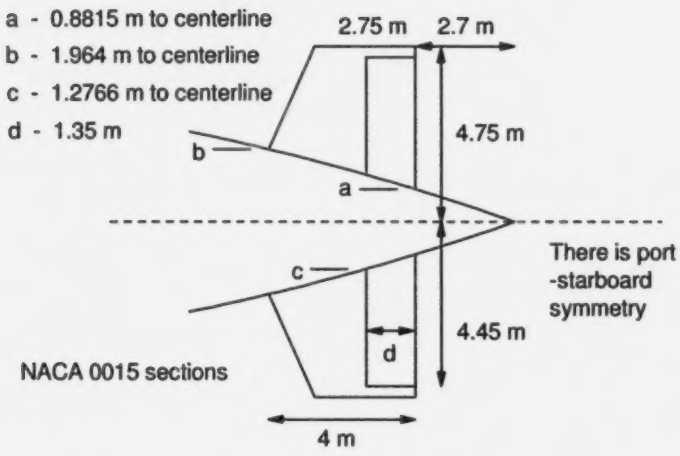


Figure A.4. Sternplane dimensions.

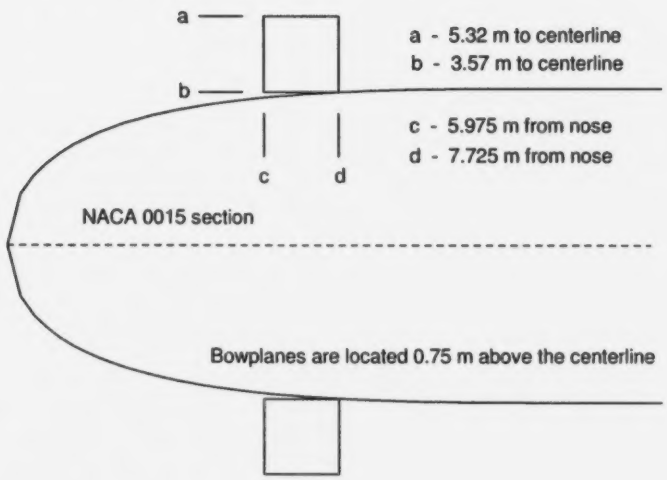


Figure A.5. Bowplane dimensions.

Exercise 1. Component Model – Hull

Components can be defined in any order; in this case we start with the hull. Hull dimensions are derived from existing offset tables or by digitizing drawings. In table A.1 this has been done for 31 equally-spaced stations. Note that x and the upper and lower offsets are expressed in standard coordinates (x forward, z down) with the origin at the nose. Section area is estimated by $\pi BT/4$ where B is breadth and T is height, and camber is the mean of the upper and lower offsets.

With the data in this form, it is simplest to make the vehicle axes coincide with the hull axis. The basic hull specification in *Root*`.geo` is then:

```
Hull
  Nose   0.0
  Tail -61.0
  Station     31
    0      0       0       0
    5.1640 5.4684 22.1788 -0.1522
    6.3437 6.8975 34.3658 -0.2769
    6.9018 7.6497 41.4665 -0.3740
    7.1675 8.0546 45.3421 -0.4435
    ...                             (complete with data from table A.1.)
    6.0326 6.4949 30.7728 -0.2311
    5.3722 5.5572 23.4476 -0.0925
    4.5738 4.5738 16.4300  0
    3.6373 3.6373 10.3909  0
    2.5629 2.5629  5.1588  0
    1.3504 1.3504  1.4323  0
    0      0       0       0
```

The x coordinate is not required for regular spacing. Here, defining the nose and tail takes care of that. Other parameters, including the midhull, length, and axis rotations are available to fully define the hull axis dimensions and attitude with respect to vehicle axes. Failure to provide a consistent combination of them generates an error.

When the program is run, there are a few messages in the *Root*`.log` file:

```
-- Information from GEO_L2_HUL
   Nose y,z coordinates default to zero

-- Information from GEO_L2_HUL
   Tail y,z coordinates default to zero

?? Warning from ALBONC
   Iterations maxed out - check error is OK

?? Warning from HYDRST
   CG is based on default BG and form CB
```

The first two and the last are merely pointing out the obvious in case the user has overlooked other input options; the third indicates that an internal calculation was stopped before reaching its target accuracy. The error in this case, which precedes the warning in the log, is small enough ($<< 1$) to be ignored. Axial potential flow data after the message are the output of that calculation, and appear normal. DSSP50 generates a lot of messages in *Root*.log, especially warnings. The rest of this tutorial will only mention fatal errors and the more interesting or obscure warnings.

Reviewing output file *Root*.gpr, it is evident that the data have been interpolated to 21 stations for internal calculations. This is the program default. Note that 'maximum effective dia.' is the maximum value of $\sqrt{BT}$; it is used in the calculated L/D ratio. Subsequently, and in the 'Total vehicle properties' section, default values are reported for CB and CG since they were left undefined. Default CB is calculated from the external envelope of the vehicle, in this case just the hull, while CG is derived from $\overline{BG}$ defaulting to 1/20 of the maximum effective diameter. The default $\overline{BG}$ is likely adequate, and we will not, for the moment, redefine it.

The point called the hydrodynamic reference, or simply the reference, is the origin for rotations and moments in hydrodynamic calculations. For numerical reasons it should be in the vicinity of the vehicle CB or CG and, as seen on *Root*.geo, defaults to the LCB of the hull if there is just one.

On *Root*.geo there are lists of dimensional derivatives (Y_v, etc.) for the different methods of load calculation. However, the final loads may be subject to additional corrections. Note that the program has estimated vehicle inertial properties, added mass properties, and autopilot parameters by default.

Add a couple more specifications. First, define the reference to be 2.5 m forward of midships on the hull axis by adding the following outside the scope of (i.e., before or after) the hull input:

```
Reference
    -28.0  .0  .0
```

A second row in the data block can be used to define reference axis rotations, but this is generally not necessary.

Next, label the hull and add some Text records as comments. Finally, although the additional elevation area represented by the casing (deck) and keel is accounted for in hull calculations, the effect of the deck and keel on crossflow separation is not. Apply half the corrections cited in reference [2]:

```
Deck  1.1  1.0
Keel  1.35 0.95
```

Static simulations give hydrodynamic loads on the hull at incidence. Create a *Root*.sim file (*Root* is the same as for *Root*.geo) containing the following:

```
Static
  Rho  1025.2
  Loop  Alfa  -20   20   1

Static
  Rho  1025.2
  Loop  Beta    0   20   1
```

Run the program. The `Loop` records cause loads to be estimated at angles of attack, α, from −20 to 20 degrees in 1 degree increments, and at angles of drift, β, from 0 to 20 degrees in 1 degree increments. Total velocity defaults to 5 m/s (9.719 kt); it can be changed with a `Ums` or `Ukt` record.

The `Rho` records set water density to 1025.2 kg/m^3 (standard seawater). By default the value is 1000 (fresh water) for static and captive simulations, and 1025.2 for free and MLD simulations.

New output consists of a *Root*`.spr` file, and a `simnnn.txt` file for each of the simulations. The first is principally a summary of what was done. The `simnnn.txt` files tabulate dimensional state variable values and hydrodynamic loads. Conversions, for example to incidence and nondimensional coefficients, must be done in post-processing software.

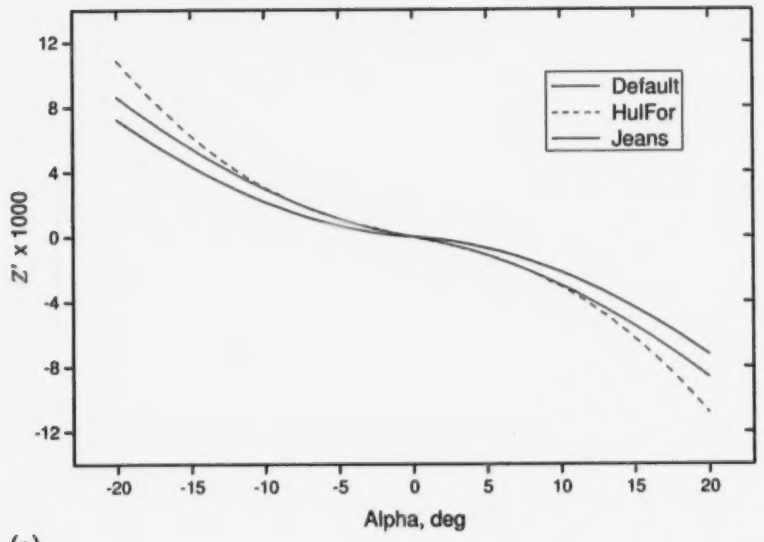
Supplementary Exercises

- Plot the hull and satisfy yourself that it is as expected; the sections are ellipses based on B and T.
- Try different ways to define hull location and attitude. The consistency requirement is illustrated in section 5.2 of reference [4].
- Omit some sections in the parallel midhull and input the rest as irregular (x is needed).
- Interpolate to different numbers of sections and try different methods for the load calculations. (See reference [1] for how to access the `HulFor` option.)

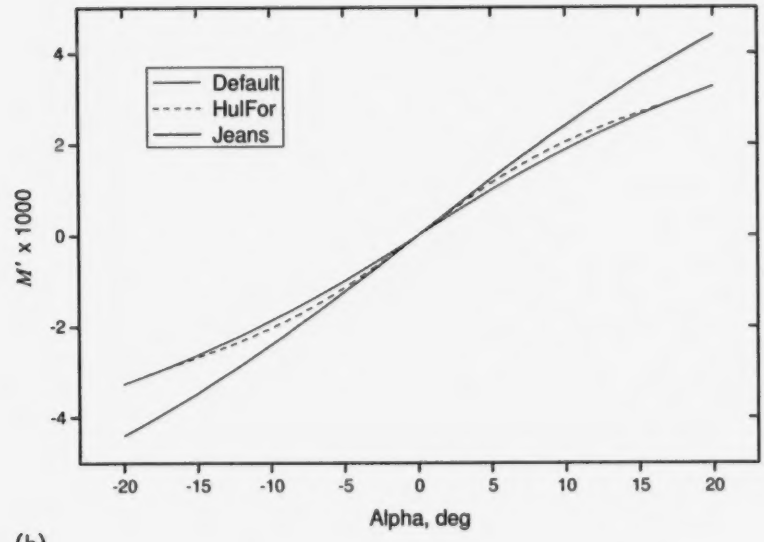
Figure A.6 illustrates differences in loads for different calculation methods. The variation seen is typical of what the author has found in a large number of similar comparisons. In this case, unfortunately, there is no data for validation. The loads are nondimensionalized in the conventional way:

$$Z' = \frac{Z}{\frac{1}{2}\rho\ell^2U^2} \qquad \text{and} \qquad M' = \frac{M}{\frac{1}{2}\rho\ell^3U^2}$$

where ℓ is the length of the hull and U is total velocity.



(a)



(b)

Figure A.6. Nondimensional loads for the hull alone in pitch: (a) normal force Z' and (b) pitching moment M'. Three calculation methods are used.

Exercise 2. Component Model – Sail

Lifting appendages, other than isolated ones, need a local hull radius, usually taken at mid root chord, to determine lift carry-over. In the case of a sail, we normally use the half-breadth, here 3.65 m. The planform definition and thickness of the sail can be taken directly from figure A.2. Add the following to *Root*.geo:

```
Lift
  Sail  3.65
  Planform
    -18.96  0.   -4.65
    -19.62  0.  -10.15
    -29.38  0.  -10.15
    -30.04  0.   -4.65
  ToC   0.18
```

Run the program. The new 'Lift' section in *Root*.geo lists the internal representation of sail geometry as well as numerous geometric and hydrodynamic parameters derived from it. There is also a 'OOP summary' section for the out-of-plane load calculations [2] associated with this sail. It shows default values of the parameters since nothing was explicitly input for them. Change the k parameter by adding the following to *Root*.geo outside the scope of the lift input:

```
OOPCalc
  OLift   1
  KParam  1.5
```

Since the sail was not labeled, it is identified by the index 1. After the program is run there is a new section 'OOPCalc' in *Root*.geo, and the value of k in the OOP summary has been revised.

Supplementary Exercises

- Plot the hull-sail configuration.
- Introduce an error into the planform specification, e.g., change `-4.65` to `-3.65` in one place. Plot and observe how much easier it is to detect the error in a wireframe rendition (with no hidden lines) than in a shaded one.
- Compare loads from the static simulations with those for the hull alone.

Exercise 3. Component Model – Rudders

Rudder geometry is sketched on figure A.3. The rudders are assumed to be completely all-moving. Add the following to *Root*.geo:

```
Lift
  Rudder  1.4228
  Label #UpperRudder
```

```
  Planform
    -54.3  0.  -1.964
    -55.8  0.  -4.6
    -58.3  0.  -4.6
    -58.3  0.  -0.8815
  ToC   0.15
  AllMoving

Lift
  Rudder  1.4228
  Label #LowerRudder
  Planform
    -54.3  0.  1.964
    -55.8  0.  4.15
    -58.3  0.  4.15
    -58.3  0.  0.8815
  ToC   0.15
  AllMoving
```

While in principle the order of components on *Root*.`geo` does not matter, this input should be entered after the sail. Otherwise, the upper rudder instead of the sail will be given the index 1, and it rather than the sail will be the object of the `OOPCalc` input. Since there are numerous instances where component indices can be referred to, new input should not change indexing unless it is unavoidable.

Alternatively, referencing components by label usually avoids indexing problems. Labels can be virtually any unique name, preferably descriptive and reasonably short.

The rudders are indicated as being completely all-moving control surfaces. There are some parameters associated with this that may differ from the default values. Add the following to *Root*.`geo` within the scope of `Lift` input for each rudder:

```
DeltaLim    30  -30
DelRateLim   6
```

Supplementary Exercises

- Run the program with the upper rudder input before the sail input. There is no error or warning, but note in *Root*.`gpr` that the `OOPCalc` x_1 vortex limit changes to a value appropriate for the rudder. Label the sail and use the label instead of the index in `OOPCalc` → `KParam`. The `OOPCalc` data should be correct again.
- Plot and check the rudders.
- See the effect of optional parameters such as tailplane efficiency on the static loads. To use Dempsey's method (not recommended for a full scale submarine), the argument $r_M(= T/2) = 4.4$ should be added to the `Rudder` record.

Exercise 4. Component Model – Sternplanes and Bowplanes

Sternplane geometry is sketched on figure A.4. The planes are flapped with the outer 0.3 m fixed for a tip bearing. The following flap parameters [1] are required:

$$\begin{aligned} b_1/b_{exp} &= 0 \\ b_2/b_{exp} &= (4.45 - 0.8815)/(4.75 - 0.8815) = 0.9225 \\ \overline{c_f}/\overline{c} &= 1.35/\tfrac{1}{2}(2.75 + 4) = 0.4 \end{aligned}$$

These are arguments to the `Flapped` record. Deflection limits and rates can be added as for the rudders. It is only necessary to input one sternplane and rotate it 180 degrees to create the other. Add the following to *Root*`.geo`:

```
Lift
  Sternplane  1.4228  3.65    !  (r_M = B/2)
  Label #StarboardSternplane
  Save                        !  (for rotation)
  Planform
    -54.3   1.964   0
    -55.55  4.75    0
    -58.3   4.75    0
    -58.3   0.8815  0
  ToC   0.15
  Flapped   0  0.9225  0.4
  DelRateLim  5
  DeltaLim   25  -25

Lift
  Label #PortSternplane
  Rotate   180
```

Run the program. The geometry should look like figure A.7.

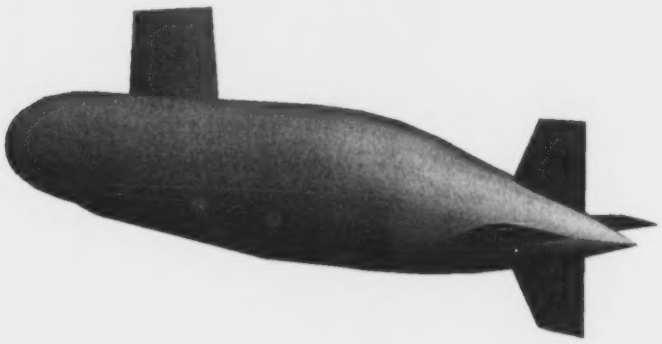


Figure A.7. Hull + sail + rudder + sternplane geometry, shaded rendition.

If they are at the end of the input file, the sternplanes are indexed 4 and 5 in *Root*.gpr. In the output sections for them, there is the comment 'no plane reversal'. This means that the program does not have data to automatically reverse the sense of control deflections below plane reversal speed, which is typically a few knots [2,5]. The reversal speed is *a priori* unknown, but it can be obtained by running a coefficient simulation. Note that even if there is empirical data for it, the value estimated by DSSP50 is the correct one to use in a simulation.

Use the root mid chord, –56.3 m, to characterize the longitudinal location of the sternplanes. Add the following to *Root*.sim:

```
Coef
  PReverse   -56.3
  CoefOut  Coef   ! Outputs coefficients to simnnn.txt file - used later
```

Run the program. Coefficient output appears as simulation 003 in *Root*.spr. The program does a series of internal static and dynamic simulations, deriving the coefficients by linear least squares regression. The coefficients are followed by nondimensional and added mass matrices, and then by stability estimates.

Sternplane reversal speed is estimated to be 4.522 kt. While this value may change with additional changes in geometry, for now we will use it as the basis of a plane reversal profile [1,2] with a deadband about 4.5 kt. Add the following within the scope of the first sternplane input on *Root*.geo:

```
  PReverse  Ukt
     3.5  4.0  5.0  5.5
    -1    0    0   1
```

The bowplanes are completely all-moving control surfaces; their geometry is sketched in figure A.5. The hull would be bumped out into a sponson or blister so that there is no root gap, at least for zero deflection. However, this feature is not modeled as it is hydrodynamically insignificant.

Add the following to *Root*.geo:

```
Lift
  SBowplane  3.5  12.4   "(phi introduces Nonweiler's correction)
  Label #StarboardBowplane
  Save                          !  (for reflection)
  Planform
    -5.975   3.57   -0.75
    -5.975   5.32   -0.75
    -7.725   5.32   -0.75
    -7.725   3.57   -0.75
  ToC  0.15
  AllMoving
  DeltaLim   22.5   -22.5
  DelRateLim  4.5
```

```
Lift
  Label #PortBowplane
  Reflected                    !  (since it is off centerline)
```

Because the bow planes are above the hull centerline the port one will not retain the correct orientation if it is rotated from the starboard bowplane (try it) — it must be reflected. The optional argument ϕ on the **SBowplane** record is the rounded-off value of $\sin^{-1}(|\text{centerline offset}|/\text{hull radius}) = \sin^{-1}(0.75/3.5)$.

Supplementary Exercises

- Run the program; plot and check the bowplanes.
- Among the coefficients output on *Root*.**spr**, it will be found that Z'_w and M'_w, in particular, have changed considerably with the addition of bowplanes. This results in a lower sternplane reversal speed of a little over 3.5 kt. Revise sternplane **PReverse** input accordingly.
- See the effect of different optional parameters for the bowplanes.

Exercise 5. Component Model – A Small Lump

This exercise illustrates the addition of a small lump to the geometry. The lump represents a mast extended from the sail. This is an artificial example, since masts are not extended for subsurface manoeuvers other than for low speed at periscope depth.

The only hydrodynamic loads are drag in the x and y directions acting at the midpoint of the mast. At this location, the corresponding moments can be assumed to be zero. The mast is modeled as a circular cylinder with diameter $d_m = 0.3$ m and length $\ell_m = 3$ m. Its hydrodynamic location is therefore $\ell_m/2 = 1.5$ m above the tip of the sail.

Reynolds number based on d_m varies from zero to a few million. Above a critical value of about 350,000 the major component of transverse drag, pressure drag, can be approximated by $\frac{1}{2}\rho U^2 d_m \ell_m C_{Do}$, where $C_{Do} \approx 0.3$ [6]. For a reference length ℓ, the coefficients for input into **DSSP50** [1] are $X'_{uu} = Y'_{v|v|} = -1000(d_m \ell_m/\ell^2)C_{Do}$, including the scaling factor 10^3.

Add the following to *Root*.**geo**:

```
Lump
  Label #Mast
  Location  -22.06  0  -11.65     !  at sail tip quarter-chord
  Length     61                   !  _not_ l_m !!
  Volume      0.2121
  Coef
    Xuu   -0.0726
    Yvnu0 -0.0726
```

The geometry of the lump does not appear explicitly in the input. Run the program. On *Root*.`gpr` output note that the lump's moments of inertia and some other parameters have defaulted to zero. It has been given non-zero default values for the control parameters, but as the corresponding control coefficients default to zero these parameters have no effect. Compare loads, coefficients, etc., with and without the lump. There is very little difference, and for hydrodynamic calculations the lump can be omitted.

For plotting, we can represent the mast as a cylinder with its actual dimensions. Add the following to *Root*.`geo` within the scope of the lump input:

```
Visualize  Cylinder  Z  3.0  0.15
```

The geometry should look like figure A.8.

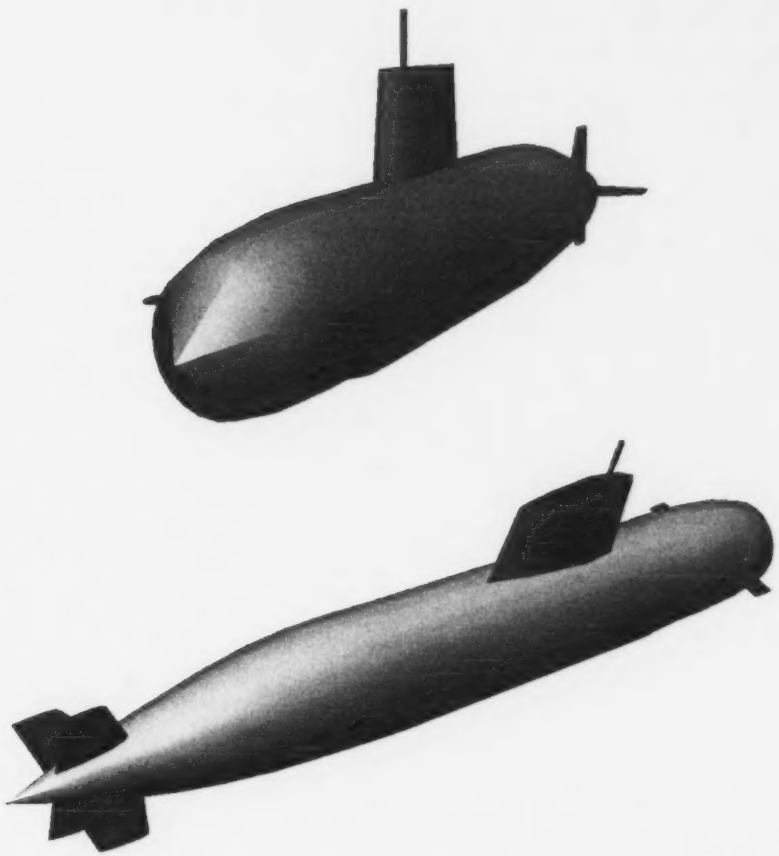


Figure A.8. Completed geometry.

Supplemen ry Exercises

- Test any untried options for the areas covered so far.
- Comment your input to make it easier to refer back to.

Exercise 6. Component Model – X-rudders and Sailplanes

Save the geometry input so far with some notation (i.e., `Text ...`) like '`+ Rudder Version`'. Various systems, e.g., propulsion, will be added in a later exercise. For the present exercise we will modify the geometry to be a X-rudder version, with sailplanes, of the same submarine.

Defining any configuration other than +-rudders is made easier by specifying an appendage in either the vertical or horizontal plane, killing it [1], then rotating it to the required position(s). For this exercise, the X-rudders are all-moving appendages with slightly larger span than the previous upper rudder, situated so that they are a little more effective in the horizontal plane than the vertical. Replace rudder and sternplane input in *Root*`.geo` with the following:

```
Lift    !! X-rudder in the upper rudder position; it will be killed
  Tail  1.4228   ! (responds to commands in both h- and v- planes)
  Kill
  Planform
    -54.3  0.  -1.964
    -55.8  0.  -4.8
    -58.3  0.  -4.8
    -58.3  0.  -0.8815
  ToC   0.15
  AllMoving
  DeltaLim    30  -30
  DelRateLim   6
  PReverse  Ukt            !! used for vertical plane when rotated
     2.5  3.0  4.0  4.5
    -1    0    0    1

Lift
  Label #StbdUpper
  Rotated   47

Lift
  Label #StbdLower
  Rotated  133

Lift
  Label #PortLower
  Rotated  227

Lift
  Label #PortUpper
  Rotated  313
```

Run the program and compare the coefficient run data on *Root*.spr with that for the +-rudder configuration. It should be found that values of the major coefficients are quite similar, as are the stability characteristics.

Note in *Root*.gpr that the starboard upper X-rudder takes the index 2 after the 'X-rudder in the upper rudder position', which is initially given that index, is killed.

Sailplanes will be added halfway up the sail at maximum thickness. They have about 15 percent greater area than the bowplanes in the +-rudder version. Replace bowplane input in *Root*.geo with the following:

```
Lift
  Sailplane  0.8  0.5
  Label #StarboardSailplane
  Save                          !  (for reflection)
  Planform
    -21.35   0.95  -7.4
    -21.85   3.3   -7.4
    -23.10   3.3   -7.4
    -23.10   0.95  -7.4
  ToC  0.15
  AllMoving
  DeltaLim   20   -20
  DelRateLim  4

Lift
  Label #PortSailplane
  Reflected
```

Run the program and again compare the coefficient run data on *Root*.spr with that for the +-rudder configuration. It should be found that there are larger changes in some of the coefficients for vertical motion, there is somewhat greater vertical stability, and the sternplane reversal speed is now a little more than 4 kt. Adjust the sternplane reversal profile to the new value:

```
PReverse  Ukt            !! used for vertical plane when rotated
   3.0  3.5  4.5  5.0
  -1    0    0    1
```

Since sailplanes may be subject to reversal, check this by altering the PReverse record in the coefficient simulation to the sailplane mid root chord location:

```
PReverse   -22.225  0  -7.4
```

The resulting sailplane reversal speed should be about 13.2 kt. In this case, we will opt to not use the sailplanes for control at speeds above 12.5 kt. Add the following within the scope of the first sailplane input on *Root*.geo:

```
PReverse  Ukt              !! not used above 12.5 kt
  12.0 12.5 13.5 14.0
   1    0    0    0
```

The geometry of the X-rudder version should look like figure A.9.

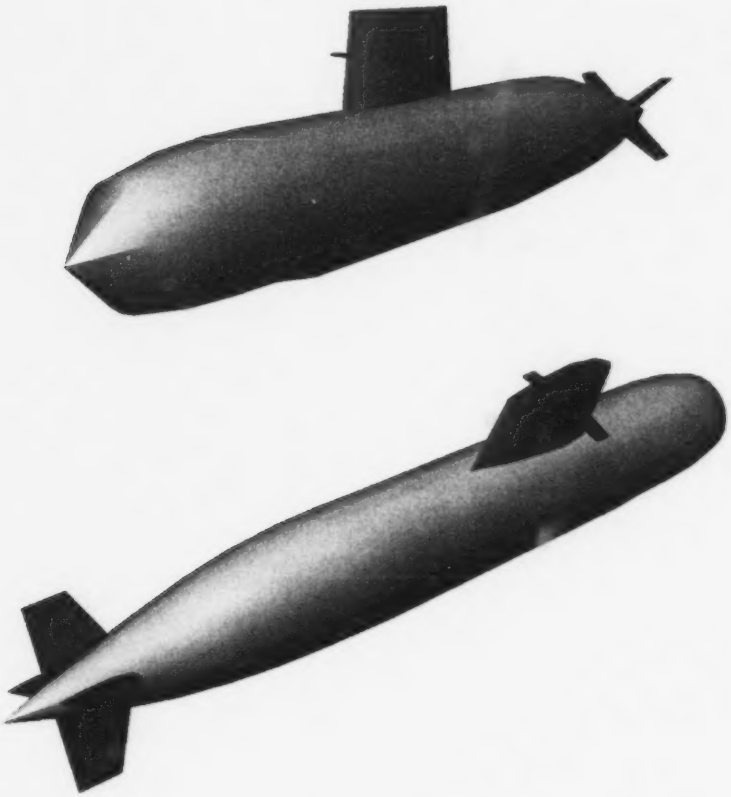


Figure A.9. X-rudder version geometry with the mast omitted.

Supplementary Exercises

- See the effect on the hydrodynamic coefficients and stability of setting the X-rudders uniformly around the hull (45°, 90°, etc) rather than as suggested above.
- Similarly try variations of sailplane dimensions and parameters.

Captive simulations provide another measure for comparing variations in geometry and other parameters. This option was implemented chiefly for predicting loads in an exper-

iment where a model of the vehicle is forced through a predetermined trajectory. Such experiments are done with test rigs like a PMM, rotating arm, or the MDTF [7].

The minimum specification of trajectory is the time history of axial speed u (in m/s). To this can be added time histories of the other state variables v, w, p, q, r, ϕ, θ, or ψ (in m/s, deg/s, or degrees as appropriate). A simple, but clearly unrealistic at full scale, example is pure harmonic heave. For $u = 5\,\text{m/s}$ and $w = 1.25\sin(0.2\pi t)\,\text{m/s}$ create a file with three columns: time sampled at 5 Hz for 40 seconds, u and w. Subject to variation in spacing, precision, and alignment, the first few lines of the file should contain:

```
time u w          !  Write identifiers as shown, but case is immaterial.
0 5 0             !  The first column must be time, the others may be
0.2 5 0.15667     !  in any order.  See reference [1].
0.4 5 0.31086
0.6 5 0.46016
0.8 5 0.60219
1 5 0.73473
1.2 5 0.85568
1.4 5 0.96314
...
```

This example was created in Origin; a spreadsheet would work as well. The comments were added after. Add the following to *Root*`.sim`:

```
Captive
  Rho  1025.2
  TimeHistry   DataEx6c.txt  ! substitute your time history file name
```

Run the program. The captive calculation is summarized as simulation 004 in *Root*`.spr`. State variable and load time histories are listed in `sim004.txt`, and acceleration time histories in `aux004.txt`.

Supplementary Exercises

- Compare captive simulation results for the +-rudder and X-rudder versions of the submarine.
- Try captive simulation options (see reference [1]).
- Try more elaborate time history files, including examples of inconsistent Euler angle input.

The last step in building a component model is adding submarine systems, exercise 9.

Exercise 7. Coefficient Model – the whole Vehicle

Some familiarity with a coefficient model of the hydrodynamic loads is assumed, but it will be useful to review exercise 5. The user who skipped the component model exercises should at least refer back for an introduction to some of the simulation modes: static

simulation at the end of exercise 1, coefficient simulation in the middle of exercise 4, and captive simulation at the end of exercise 6.

The primary requirement is to provide values for a representative set of the coefficients listed in annex B of reference [1]. If a simple lump is being modeled, as few as two coefficients may be sufficient as illustrated in exercise 5. However, modeling a whole vehicle requires at least the major linear damping coefficients (Y'_v, Z'_w, etc.) and inertial properties.

For this exercise, we will use the predicted coefficients obtained in one of the previous exercises, but the user can substitute any available data. If using other data, note that the coefficients must have been nondimensionalized by integer powers of a reference length ℓ (i.e., ℓ^2, ℓ^3, etc.), and that they are multiplied by 1000 for input.

The `sim003.txt` output file from exercise 5 is:

```
LLRS                        COEFx1000
Xuu                -0.125971462621020E+01
Yuu                -0.738683288647072E-47
Zuu                -0.603200601026944E-31
Kuu0                0.199346311441066E-32
...
Yu1r1dr             0.103979810427884E+02
Nu1r1dr            -0.464638309614398E+01
Zu1q1ds            -0.928619572379280E+01
Mu1q1ds            -0.418253260133347E+01
```

This list of 94 coefficients does not include added mass and moment coefficients, which have to be extracted from the table in *Root*`.spr`, or propulsion-related coefficients Y'_{vn}, etc., which are generally available only from model experiments. After omitting or commenting out the header, the list can be put directly into *Root*`.geo`.

In this case, inertial properties (which can be extracted from *Root*`.spr`), the reference, CG, etc., are already known from the preceding coefficient run. If independent data are used, make sure the reference is correct for the coefficient values.

Create a new *Root*`.geo` file as follows, using all the coefficient values available:

```
Text Tutorial exercise 7 file a
Text Whole-vehicle coefficient model derived from exercise 5.
Text (or use your own data !)

! Plot   !  not very interesting ...

Reference
    -28.0  .0  .0

CG    -27.095411       0.000000       -0.111290       ! from ex5 gpr file

Lump
```

```
Label #OneLump
Length      61.0
Location   -28.0 .0  .0  ! at ref.
MCB         0.904589        0.000000       -0.512040 ! from ex5 gpr -
MCG         0.904589        0.000000       -0.111290 ! - but wrt ref.
MomInertia
     19234025.7    452179413.3   545178885.9        ! from ex5 gpr file
Volume             2443.2756                         ! from ex5 gpr file
Bowplane                          ! values from ex5 geo file
   DeltaLim    22.5  -22.5        !   "
   DelRateLim   4.5               !   "
Rudder                            !   "
   DeltaLim    30.   -30.         !   "
   DelRateLim   6.                !   "
Sternplane                        !   "
   DeltaLim    25.   -25.         !   "
   DelRateLim   5.                !   "
   PReverse  Ukt                  !   "
      2.5  3.0  4.0  4.5          !   "
     -1    0    0    1           !   "
Coef
   Xuu                 -0.125971462621020E+01
   Yuu                 -0.738683288647072E-47  ! (zero -- can ignore)
   Zuu                 -0.603200601026944E-31  !     "
   Kuu0                 0.199346311441066E-32  !     "
   ...                                         !! Use complete list !!
   Yu1r1dr              0.103979810427884E+02
   Nu1r1dr             -0.464638309614398E+01
   Zu1q1ds             -0.928619572379280E+01
   Mu1q1ds             -0.418253260133347E+01
   Yuudrn          2.55   !  similar geom/guesstimate
   Zuudsn         -1.85   !    "
   Xudot       -0.77167   ! derived from ex5 gpr file
   Yvdot      -23.9204    !   "
   Zwdot      -17.5331    !   "
   Kpdot       -0.03048   !   "
   Mqdot       -0.88173   !   "
   Nrdot       -1.09107   !   "
```

Run the program. There is one warning that is superfluous since $\overline{BG}$ is defined implicitly by MCB and MCG records in the lump input:

```
?? Warning from GEO_L1
   BG is undefined
   - there is no default if not just one hull
```

It is followed by two warnings also related to the absence of a hull because there is no hull length to use as a reference. Enter/Return to ignore each one. They are:

```
?? Warning from GEO_L1
```

```
   Autopilot depth error gain (CZZ)
   - could not be initialized (no hull)

?? Continue execution ? ... [Y]/N

...

?? Warning from GEO_L1
   Autopilot depth LOOKAHEAD
   - could not be initialized

?? Continue execution ? ... [Y]/N
```

Adding a record containing **Warnings Quiet** at the top of *Root*.**geo** will cause the warnings to be automatically ignored [1].

Finally, if the program was run with the *Root*.**sim** file developed in the previous exercises, it will stop with a fatal error:

```
!! Fatal Error in SIM_L2_COF
   Reference length undefined
   - there is no default if not just one hull
```

To fix that, add a record containing **Length 61.0** within the scope of the coefficient simulation input on *Root*.**sim**. Run the program with all the simulations. The coefficients and stability characteristics should have been reproduced within roundoff accuracy.

Supplementary Exercises

- Write the coefficient model to a file (**Save** record); the result can be read if it is in ASCII format. Replace the lump data with this file (**File** record), and check that the resulting output is correct. **Coef** and other *Root*.**geo** data should be able to overwrite what is on the file — try it.
- Omit the coefficients associated with nonlinear terms in the equation of motion except those for axial and translational drag. Compare the output from this linearized model with that of the full coefficient model. Loads at other than small velocities or incidence will have likely changed significantly, although the stability characteristics, which are based on linearization, should be unchanged within roundoff accuracy.

Exercise 8. Hybrid Model – Modified Appendages

Any model that combines component and coefficient model elements can be termed a hybrid, and that general definition includes the configuration developed in exercise 5. However, insofar as a lump is treated as any other component, the distinction is an artificial one, so we tend to use the term hybrid when both lumps and other components are hydrodynamically significant.

Suppose that the submarine represented by the full coefficient model in exercise 7 is deemed to be deficient in vertical stability, and we wish to explore the effect of increasing sternplane area by about fifty percent, keeping the flaps in proportion. Obviously, a reasonably accurate knowledge of the current sternplane dimensions is necessary. If your coefficient model is based on the +-rudder configuration of exercise 5, current sternplane geometry is defined in figure A.4. If another model was used, this example should be modified accordingly.

The procedure is to subtract loads due to the current sternplanes and add loads due to resized sternplanes; i.e., their component weights will be set to −1.0 and 1.0 respectively. In this example, the resized planes have increased chord while the span is unchanged. Add the following to the *Root*.**geo** file for the previous exercise:

```
Text Subtract current sternplanes
Lift
  Subtract
  Sternplane  1.4228
  Label #OldStarboardSternplane
  Save                          !  (for rotation)
  Planform
    -54.3   1.964   0
    -55.55  4.75    0
    -58.3   4.75    0
    -58.3   0.8815  0
  ToC   0.15
  Flapped   0  0.9225  0.4
  DeltaLim   25  -25
  DelRateLim  5
  PReverse  Ukt
     2.5  3.0  4.0  4.5
    -1    0    0    1
  TailEff    0.7  0.4

Lift
  Subtract
  Label #OldPortSternplane
  Rotated   180

Text Add new sternplanes -- note different labels
Lift
  Sternplane  1.6432
  Label #NewStarboardSternplane
  Save                          !  (for rotation)
  Planform
    -52.3   2.4049  0
    -54.175 4.75    0
    -58.3   4.75    0
    -58.3   0.8815  0
  ToC   0.15
  Flapped   0  0.9225  0.4
```

```
  DeltaLim   25  -25
  DelRateLim  5
  PReverse  Ukt
     2.5  3.0  4.0  4.5
    -1    0    0    1
  TailEff    0.7  0.4

Lift
  Label #NewPortSternplane
  Rotated   180
```

Run the program. We do not normally plot primarily-lump hybrid models, but this is done in figure A.10 to illustrate the two sets of sternplanes. For contrast, the resized planes have been rendered in blue by manually editing the plotting file *Root*`.wrl`.

Figure A.10. Hybrid model with resized sternplanes, wire frame rendition. The lump is visualized as a sphere of radius 0.1ℓ.

Note that tailplane efficiency must be input explicitly since there is no hull present [1]. The values above are rough estimates for the purpose of the exercise, but it would be better to use empirical values for similar planes, if available. It is difficult to quantify how the increased chord and reduced aspect ratio of the resized planes affect tailplane efficiency in this case, so the efficiencies are unchanged. Uncertainties in these and other parameters only apply to differences between the two sets of planes.

The sternplane location (mid root chord) for plane reversal in the coefficient run on *Root*`.sim` becomes –55.3 m instead of –56.3 m.

It should be found that the magnitudes of the principal coefficients have changed as expected, that of Z'_w by about nine percent for example. The magnitudes of the sternplane coefficients $Z'_{\delta s}$ and $M'_{\delta s}$ have increased 20 to 25 percent. The vertical plane stability index G_v has increased by about 20 percent.

The next example consists of lengthening the sail by moving its trailing edge aft by 2 m. The basic procedure is as before. Add the following to the *Root*.geo file for the pure coefficient model in exercise 7 — with appropriate modifications if the submarine is not the +-rudder configuration developed above:

```
Text Subtract current sail
Lift
  Subtract
  Label  #OldSail
  Sail  3.65
  Planform
    -18.96  0.   -4.65
    -19.62  0.  -10.15
    -29.38  0.  -10.15
    -30.04  0.   -4.65
  ToC   0.18

Text Add new sail -- note different label
Lift
  Add    ! redundant!
  Label  #NewSail
  Sail  3.65
  Planform
    -18.96  0.   -4.65
    -19.62  0.  -10.15
    -31.38  0.  -10.15
    -32.04  0.   -4.65
  ToC   0.15
```

Note that the thickness/chord ratio of the resized sail has been reduced to maintain the same maximum thickness. In order that out-of-plane load calculations [2] reflect the change in the sail (although the effect will be small), the following should also be added to the *Root*.geo file:

```
Text Correct out-of-plane loads for new sail trailing edge

OOPCalc
  Subtract
  OLift  #OldSail
  KParam  1.5
  XoneXtwo
         -30.04  0  0
         -61.    0  0

OOPCalc                    ! add by default
  OLift  #NewSail
  KParam  1.5
  XoneXtwo
         -32.04  0  0
         -61.    0  0
```

Run the program. The predicted sternplane reversal speed is a little higher than before, about 3.7 kt. Adjust the lump data accordingly. Note changes in the estimated yaw and roll coefficients. The horizontal plane stability index is substantially increased.

Figure A.11 illustrates the change in sail geometry. For contrast, the resized sail has been rendered in orange by manually editing *Root*.`wrl`.

Figure A.11. Hybrid model with resized sail, wire frame rendition. The lump is visualized as a sphere of radius 0.1ℓ.

Supplementary Exercise

- Try another modification of your coefficient model. A simple example would be to replace bowplanes with sternplanes or *vice versa*, depending which configuration your model corresponds to. A more complicated example would be to add a plug (section of hull) to the forebody or afterbody. In that case, appendages may have to be moved and out-of-plane load and other corrections made, utilizing much of the vehicle geometry. It is most likely more effective to use a regular component model of the submarine to study the effect of hull plugs and similar modifications than to work with a hybrid model.

Exercise 9. Propulsion, Autopilots, MBTs, and Other Systems

This exercise applies to both component and coefficient based models, and can be done with any version of the submarine developed above. It is not necessary to include a *Root*.`sim` file when running the program.

We will assume the component model from exercise 5 is used. It will have a single propeller with diameter 0.55 of hull breadth, located midway between the tail appendage trailing edges and the tail. Add the following to *Root*.`geo`:

```
Prop          ! Label not required
    Location   -59.65
    Diameter     4.015
```

Propulsors default to a Wageningen B propeller. Run the program. If the submarine is based on a purely coefficient model or if there is not a single hull component, an error is generated because a reference hull diameter is needed to initialize a Wageningen B propeller. It is best to use the effective diameter $\sqrt{BT}$ if it is known. Add the following to *Root*.geo and run the program again:

```
WagBInit
    Diameter  8.015  ! (use appropriate value; this is effective dia)
```

Root.gpr now has an input section for the propeller and another called '**Wageningen B initialization**'. The latter contains estimates of characteristics for an optimized propeller [2]. Pitch/diameter ratio is generally somewhat low. It can be increased by increasing propeller diameter or by explicitly forcing a higher value, the result being somewhat less optimized. Force the pitch/diameter ratio to 0.65 by adding the following within the scope of **WagBInit** input on *Root*.geo:

```
PoD   0.65
```

For this propeller, RPM/kt is estimated to be about 11.0 and the advance ratio is 0.7 at the default initialization speed of 5 kt. For a maximum speed of about 25 kt, set maximum RPM to 270; a reasonable rate of change is 1/20 of that per second. Add the following within the scope of the Prop input above:

```
RevLimit   270.0  ! or try RevGroups; see [1]
RevRateLim  13.5
```

Supplementary Exercises

- Observe the effect of different pitch/diameter ratios on the wake fraction and other parameters (which are estimated for the default initialization speed of 5 kt).
- Try an older propulsion arrangement of two smaller type DSSPtwo propellers, one in the 0430 position and the other in the 0730 position below the sternplanes. They should be sized to fit with adequate clearances, but the program will not check that. Include shaft angles (Pitch/Yaw). Rotations should be opposite.

Autopilot characteristics are listed on *Root*.gpr. There is little reason to change the default values unless free simulations suggest otherwise (an excessive overshoot when changing heading or depth, for example). We will jump ahead and assume that has happened. Add the following to *Root*.geo:

```
AutoDepth
   DepConGain  25.0  " Sternplane limit
   DepErrGain   0.1  " Reduced to dampen autopilot response
   LookAhead   61.0  " Increased to L to reduce overshoots
```

```
AutoProp
   UktErrGain  0.05
```

These parameters are indicated to be input, instead of defaults, on *Root*.gpr. Their values may have to be adjusted later.

Compression (loss of buoyancy) of the hull and of hull tiles or cladding can be modeled. The loss and its location are specified for a series of depths. Compression continues for a hull below the last depth given (in this example, 300 m), but ceases for tiles. Add the following to *Root*.geo:

```
Compress
   CHull
     300.     6000.    0.9
   Tile
     200.     8000.    1.2
     300.    10000.    1.4
```

The information is summarized in a 'Compression data' section on *Root*.geo. The values above are arbitrary and buoyancy loss may be somewhat large for a submarine of this size; the total loss is 16 t at 300 m. Note that compression makes the submarine unstable with respect to changes in depth.

The ballast blowing system, comprising HP air bottle groups and Main Ballast Tanks (MBTs), is described in some detail in references [1] and [2]. Add the following to *Root*.geo to create two MBTs, one forward and one aft:

```
MBT
   Label #FwdMBT
   Fwd
   Location  -8.0  0   0.3
   Massflow  20.0
   Vent      24.0
   Volume   130.0

MBT
   Label #AftMBT
   Aft
   Location -53.5  0   0.
   Massflow  20.0
   Vent      12.0
   Volume    90.0
```

The total MBT volume is about nine percent of the submarine's form displacement, a rather low margin of buoyancy. In practice the forward and aft tank volumes would be further subdivided for safety and better control of blowing. The blowing (MassFlow) and venting rates are arbitrary, but not untypical in relation to the tank volumes.

Running the program at this point generates an error because there is no HP air specified for blowing ballast. Add the following to *Root*.geo to create air bottle groups for forward and aft blowing, together with a backup bottle group:

```
HPAir
   Label #FwdBlow
   Fwd
   Pressure  300.
   Volume      1.5

HPAir
   Label #AftBlow
   Aft
   Pressure  300.
   Volume      1.0

HPAir
   Label #Backup
   Main
   Pressure  250.
   Volume      2.0
```

Each of the MBT and HP air inputs gets an output section in *Root*.gpr. If simulations were included in the run, MBT-bottle group associations are listed in the configuration summary at the top of *Root*.spr.

Flooding simulations require specification of watertight compartments within the pressure hull. For this exercise we assume three compartments of roughly equal length. Add the following to *Root*.geo:

```
WTC
   Label #FwdComp
   Diameter    6.8
   Length     12.0
   Location  -18.0   0  0

WTC
   Label #MidComp
   Diameter    7.0
   Length     12.0
   Location  -30.0   0  0
   Occupancy   0.2

WTC
   Label #AftComp
   Diameter    6.2
   Length     14.0
   Location  -43.0   0  0
   Occupancy   0.4
```

Differences in diameter reflect domes forward and aft, and an aft taper in the pressure hull. However, `Diameter` and `Length` values are not particularly critical since the floodable volume of each of these compartments exceeds the reserve of buoyancy, and a flood exceeding, or even approaching, that amount is likely unrecoverable.

The final system to model is trim and compensation. This is described in some detail in references [1] and [2]. For this exercise we will model forward and aft trim tanks and a compensation tank at roughly midships. Add the following to *Root*`.geo`:

```
Trim
   Label #TrimFwd
   Location  -14.5 0 0
   Volume     12.8

Comp
   Label #CompMid
   Volume     41.0
   Location  -32.2 0 0

Trim
   Label #TrimAft
   Location  -46.5 0 0
   Volume     11.4
```

Running the program at this point generates a fatal error since neither the trim nor the compensation flow rates have been specified. Add:

```
CompRate  0.10

TrimRate  0.01
```

The tank volumes are arbitrary and somewhat low. It is not necessary for forward and aft trim tanks to have the same volume although they should be similar. The compensation tank is based on the estimated compression buoyancy loss plus 25 t.

This completes the definition of submarine systems.

Supplementary Exercises

- Try any unused options in *Root*`.geo` input.
- Add systems to other versions of the submarine for which there are input files.
- Complete descriptive comments on the files as required.

Exercise 10. Free Manoeuvering Simulations

Free manoeuvering simulations and MLD curve calculations in the next section can be run with any of the submarine geometries developed to this point. The simulation described below was done with the +-rudder component-based model from exercise 5

with systems added. The dynamics will be slightly different if another version of the submarine is used. This simulation can be added to the current *Root*.**sim** file or put on a new one.

The simplest free simulation is to let the submarine run with neutral buoyancy and trim, which are the default initial conditions, without invoking any control commands. For this we need to specify initial speed or RPM, and start and stop times. Create a *Root*.**sim** file containing the following:

```
Text Minimal simulation -- straight ahead for 60 sec.
Free
   Ukt  6  !  Initial speed or RPM is mandatory
     0     Start
    60     Stop
```

Root.**sim** records that begin with a number are commands. **Start** and **Stop** are null commands because they have no control action associated with them. The simulation runs from the earliest command (in time) to the latest. Time can be negative. Other records set simulation parameters and initial conditions. Within the scope of **Free**, records, including commands, can be in any order unless they are part of a data block, but it is recommended that the user develop a style that keeps *Root*.**sim** readily understandable as more input is added.

Run the program. If the *Root*.**geo** file contained compression data (as in the previous exercise), a fatal error is generated indicating an unexpected stop while initializing the simulation. Above that is a warning that routine DYDT called a stop because of zero or negative depth. Normally this action is benign, but it is not allowed during initialization. There are two things going on here. The first is that depth defaults to zero in a free simulation, and the second is that zero or negative depth causes a stop when buoyancy (because of compression) or mass (because of flooding and blowing) may be changing — conditions that make the dynamics of the vessel depth-dependant. A quick, but not very satisfactory, work-around is to temporarily comment out compression data so the simulation will be run at zero depth.

Since in any case we want to do this exercise with positive depth, do not suppress compression but add the following record (all additions in this section must be within the scope of the **Free** input) to *Root*.**sim**:

```
Depth  100
```

This time the submarine will run straight ahead at a depth of 100 m for 60 s. The trajectory of the vehicle is tabulated on *Root*.**spr**. Observe that the timestep rapidly increases to its default maximum value of 2 s. The file also lists a number of parameters including propulsion conditions at the initial speed of the simulation. Time histories of the principal state variables are output on a **sim***nnn*.**txt** file and time histories of systems variables are output on an **aux***nnn*.**txt** file.

By default the initial heading is along the chart x axis (x_e), or compass North [1]. Change this by adding:

```
Heading NE
```

On *Root*.spr the initial heading is now reported as 45 degrees and the x and y components of the trajectory are adjusted accordingly.

There are many initialization and other parameters that can be set, but first modify the simulation to make the manoeuver more interesting. Change initial depth and speed to 65 m at 4 kt. Replace Start with an autopilot depth change to 110 m, and increase RPM to 82 (about 7.5 kt) for greater control authority. A run time of 300 s is needed. *Root*.sim should look something like:

```
Text Exercise 10
Free
   Ukt      4
   Depth    65
   Heading NE
     0      AutoDepth 110
     0      Prop       82
   300      Stop
```

Run the program and observe or plot the output that is on *Root*.spr, sim*nnn*.txt, and aux*nnn*.txt. It is to be understood that numerical values in the following and similar descriptions can be qualified as approximate in nearly every case.

The depth change is complete after 150 s with no overshoot, and final speed is reached after 250 s. The submarine pitches –7 degrees (i.e., down) during the depth change and adopts a final attitude of –1.4 degrees with the sternplanes and bowplanes at one degree to rise. The final attitude is not level because of buoyancy loss and also because attitude is speed-dependent. The compensation/trim system can correct for both effects but that requires some trial-and-error.

Note that the ratio of real to CPU time on *Root*.spr is about 1/6 of that for the straight-ahead run, indicating shorter, and therefore more, timesteps under autopilot control; this increases calculation time. The steps are tabulated on aux*nnn*.txt. Data on the same file include the autopilot error and control signals, and the propulsion ratio n', which will correspond inversely with the increase in axial velocity u.

The submarine descends to 90 m in 107.9 s, at which time start a flood from a 80 mm breach in the forward compartment by adding:

```
107.9   Flood   1   0.08  ! (could use the label #FwdComp)
```

Now the submarine continues to descend past 110 m with the autopilot attempting to regain that depth. At 165.6 s the submarine is at 112 m and has taken on 8 t of floodwater. Recover by manually setting the planes to full rise:

```
165.6   AutoDepth   Stop     "Del Up" will be ignored without this
165.6   Del   Up   25
```

The submarine rises to 68 m at 275 s before starting to descend again. At 210 s, pitch momentarily exceeds 25 degrees; ease off the planes before this happens. Turning the submarine will help it continue to rise as will a short blow of the forward MBT:

```
178.5   Del   Up    5      ! ease off the planes
178.5   Del   Rudder -20   ! turn to stbd
178.5   Blow  1  1         ! (same as Blow Fwd Fwd)
188.5   Blow  Stop         ! secure blow after 10 s.
```

The submarine now rises too quickly because additional buoyancy is gained from expansion of air in the MBT. At 200 s vent the MBT; at 210 s the flood is assumed to be secured and RPM is increased to maintain control authority with the 14.2 t of floodwater in the forward compartment:

```
200.    Vent  1            ! vent
210.    Flood Stop         ! secure flood
210.    Prop  115
```

By 240 s the submarine is rising on a heading reciprocal to its original course. Use the autopilots to resume the original course at a depth of 20 m:

```
240.    AutoDepth  20
240.    AutoHead  Stbd  0  ! force continued turn to stbd
```

Also increase the duration of the simulation to 600 s. Run the program. Once final depth is achieved, the submarine proceeds with a pitch angle of 4.3 degrees bow up. The depth is greater than called for because by default the autopilot tries to balance both commanded depth and a pitch angle of zero. Floodwater, which here is largely responsible for the pitch angle, cannot be pumped out, but within limits it can be countered by the trim/compensation system. Add the following to *Root*`.sim`:

```
495.    Trim   7100  1    3
495.    Comp  14200  Out  2
```

Run the program. At 600 s, discrepancies in both depth and pitch are now smaller although the trim and compensation transfers are not yet complete. Indeed, the trim transfer cannot be completed as called for since there is only capacity for 5.7 m^3 of water to be pumped aft — see the tank volume specifications above, and recall that by default the tanks are initially half full. In this situation, `DSSP50` does as much of the transfer as it can.

Look at the end of the *Root*`.log` file. There is a list of target volumes for each trim or compensation transfer, representing the limits of capacity, with target times for completing each transfer. In this example, it says that the trim will be complete at 1065 s and the compensation will be complete at 633.5 s. Confirm this by increasing the simulation stop time to 1200 s and examining the resulting trim/compensation time histories. By the end, depth is 20.4 m and pitch is less than one degree.

This example is not a realistic operational manoeuver in many ways. Rather, it is a composite manoeuver that uses many of the free simulation control commands available. The user is encouraged to try others as suggested in the supplementary exercises at the end of the section; however, we will first review the results so far and describe some other input requirements.

Observe that there are oscillations in heading after the original course is regained (there is no intrinsic restoring action for horizontal plane perturbations). They are easiest to see on a time history of azimuth or ψ. Reducing the rudder angle via the heading control gain (`AutoHead` → `HedConGain` on *Root*`.geo`, see reference [1]) alleviates the problem, but results in a larger turning circle. It is more effective to increase the phase lead autopilot amplitude gain and time constant by adding the following to *Root*`.geo`:

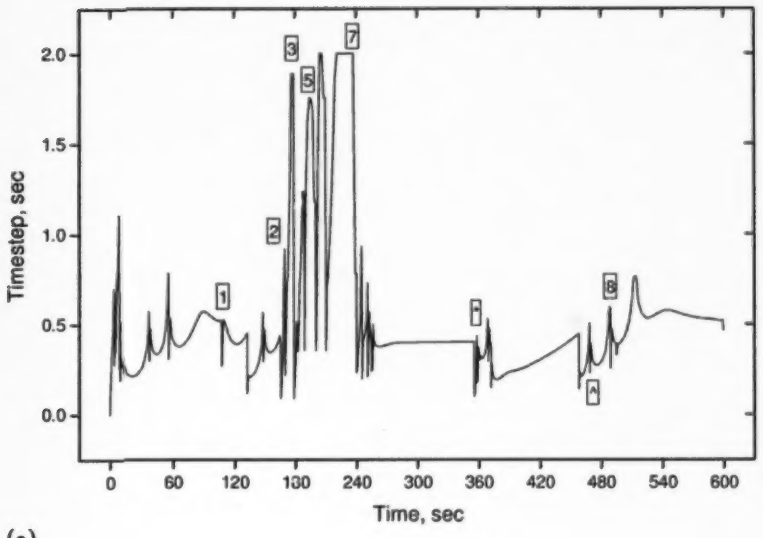
```
AutoHead
   PhaseLead     10     30
```

Both these parameters are now many times greater than normal, and minor instabilities eventually result. It is advisable to turn off the heading autopilot at about 600 s. The course subsequently deviates by only a very small amount, and can be manually controlled with rudder commands.

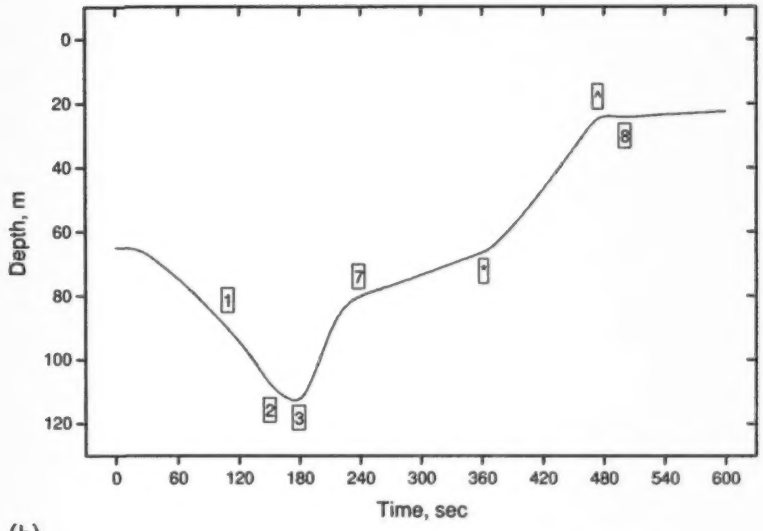
Figure A.12 shows time histories of some of the state variables and other key parameters. The figure is labeled for the events listed in table A.2, each numbered event comprising a command or bundle of simultaneous commands. Two other events are also labeled: when the submarine reaches final heading (*) and when it reaches final depth (^).

Table A.2. Exercise 10 command list.

Event	Command		
	0	AutoDepth 110	
	0	Prop 82	
1	107.9	Flood 1 0.08	! (could use label #FwdComp)
2	165.6	AutoDepth Stop	
	165.6	Del Up 25	
3	178.5	Del Up 5	! ease off the planes
	178.5	Del Rudder -20	! turn to stbd
	178.5	Blow 1 1	! (same as Blow Fwd Fwd)
4	188.5	Blow Stop	! secure blow after 10 s.
5	200.	Vent 1	! vent
6	210.	Flood Stop	! secure flood
	210.	Prop 115	
7	240.	AutoDepth 20	
	240.	AutoHead Stbd 0	
8	495.	Trim 7100 1 3	
	495.	Comp 14200 Out 2	
	600	AutoHead Stop	
	1200	Stop	

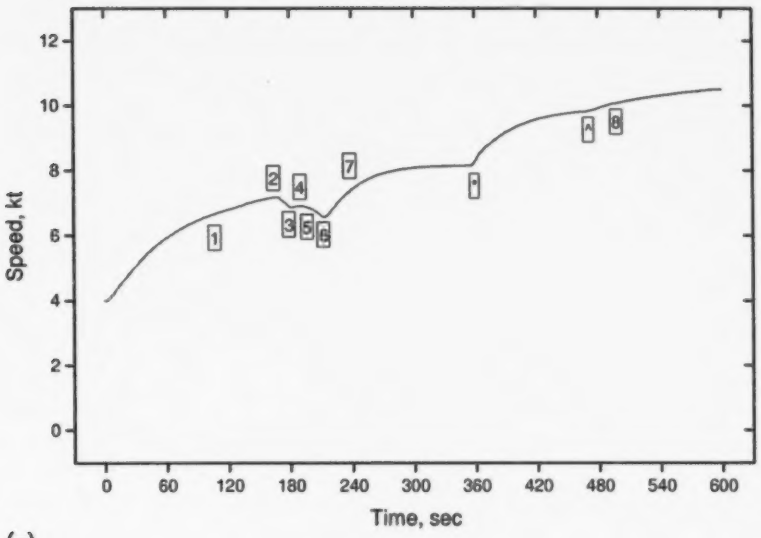


(a)

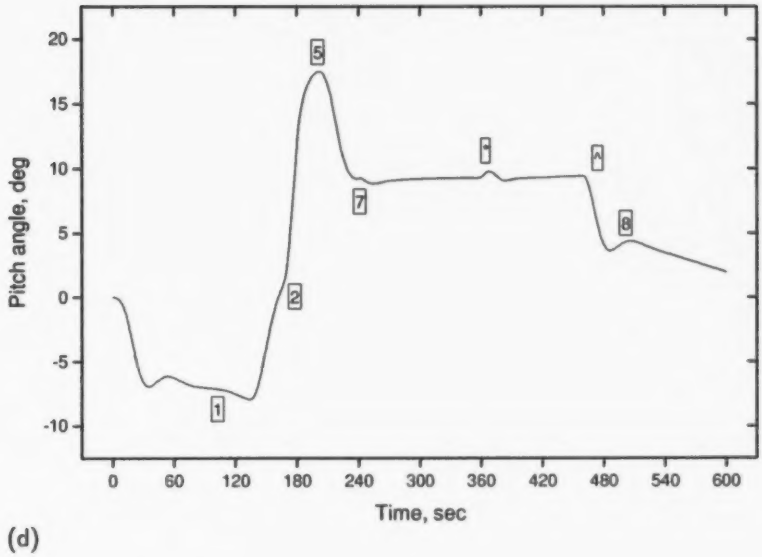


(b)

Figure A.12. Example free manoeuver time histories: (a) internal timestep size, (b) depth. The labels refer to events listed in the text. (continued)

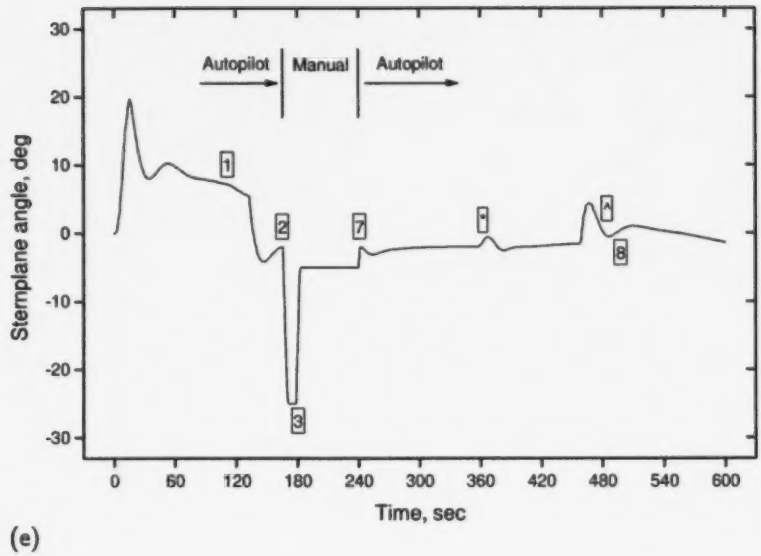


(c)

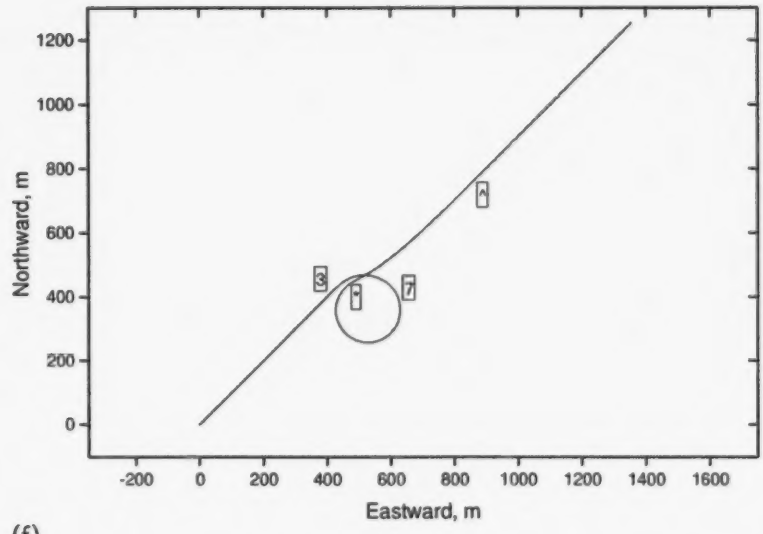


(d)

Figure A.12. (cont.) Example free manoeuver time histories: (c) speed, (d) pitch. The labels refer to events listed in the text. (continued)

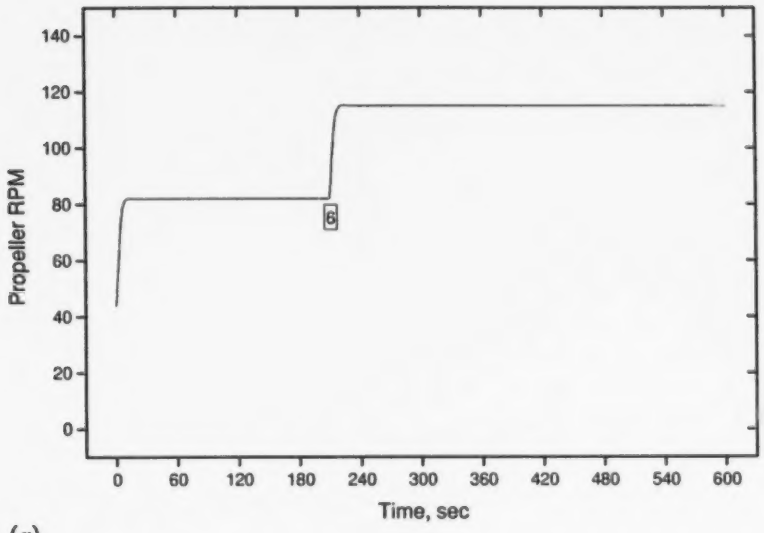


(e)

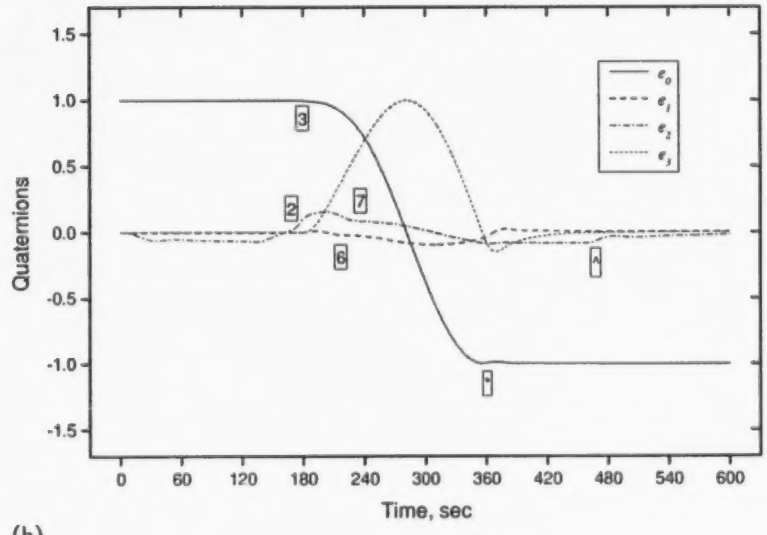


(f)

Figure A.12. (cont.) Example free manoeuver time histories: (e) sternplane angle, (f) horizontal plane trajectory in chart axes. The labels refer to events listed in the text. (continued)

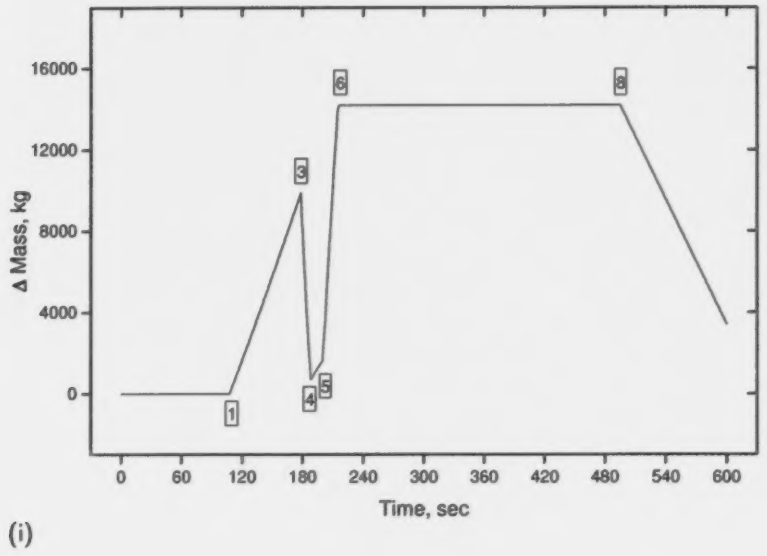


(g)

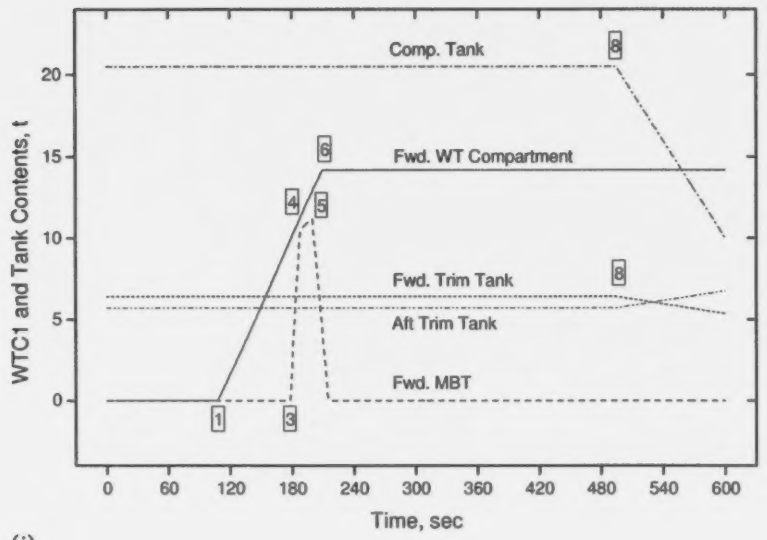


(h)

Figure A.12. (cont.) Example free manoeuver time histories: (g) RPM, (h) quaternions [2]. The labels refer to events listed in the text.
(continued)



(i)



(j)

Figure A.12. (cont.) Example free manoeuver time histories: (i) change in overall mass, (j) contents of WTC 1 and various tanks. The labels refer to events listed in the text.

There are a number of other parameters that can be set with level 3 input following an `Initialize` or `Settings` record. With a few minor differences, the following description also applies to this input in a `MLD` simulation.

`Initialize` input is used to set initial control deflections and limits, initial mass and moment imbalances, and initial dynamic conditions. The control and inertial parameters retain their initial values until changed in the simulation, but the initial dynamic conditions are treated as initial perturbations. It will be easiest to see the effect of initializing various parameters with the 6 kt straight-ahead run simulation at the beginning of this exercise.

To set the initial sternplane angle at ten degrees to dive, add:

```
Initialize
  Delta Sternplane 10
```

The only difference between this and a command '`0 Delta Sternplane 10`' (assuming 0 s is the start of the simulation) is that in the latter the sternplanes are ordered to move to ten degrees from its initial value. In both cases the submarine will dive.

This form of initial deflection specification (quasi-direct) is recognized by all types of models: component, coefficient, and hybrid. Alternative forms are:

```
Initialize
  Delta #StbdSternplane   10
  Delta #PortSternPlane  -10
```

for specific appendages in a component model, and

```
Initialize
  Sternplane #OneLump   ! (label or index)
    LDelta  10
```

for a specific lump in a coefficient or hybrid model. The deflection limits, which are initially set by default or by *Root*`.geo` input, can only be redefined in *Root*`.sim` for either a specific appendage or a specific lump. To set limits of ±10 degrees, use:

```
DeltaLim #StbdSternplane  10  -10
DeltaLim #PortSternplane  10  -10
```

or

```
Sternplane #OneLump   ! _Do_Not_ repeat this record
  LDeltaLim  10  -10
```

within the scope of `Initialize`. It is noted below that deflection limits in a MLD simulation can be made speed-dependent by using `SpeedStop` and `LSpeedStop` input. In a free simulation these are treated as `DeltaLim` and `LDeltaLim` input; i.e., the speed-dependence is ignored.

Initial mass and moment imbalances, specified relative to neutral buoyancy and trim, remain for the duration of the simulation. Still within the scope of `Initialize`, add the following to *Root*`.sim` for the straight-ahead run:

```
OutOfTrim   2000  4000  400  ! mass, pitch, roll
```

This represents an extra 2 t located 2 m aft and 0.2 m to starboard of the reference. The submarine responds to this load with respect to vertical plane dynamics as it does to control deflections. In particular, there is an associated critical speed that is analogous to plane reversal speed and is calculated the same way. A coefficient run will give a critical speed of 5.734 kt for the location of the extra load. The effect can be seen by running simulations with initial speeds of 2, 6, and 10 kt. It will be found that the submarine rises at 2 kt, dives at 10 kt, and deviates only a little in depth at 6 kt.

The other initialization inputs, **Pitch**, **Roll**, **Rotation**, **Velocity**, and **Yaw** [1] are implemented as perturbations. This can be demonstrated for an initial pitch angle of ten degrees to dive. Do a simulation with the following initialize input and no control commands (just **Start** and **Stop**):

```
Initialize
   Pitch   -10
```

The pitch angle will return to zero within 25 s, but we observe overshoot and damped oscillations. It stabilizes at about 140 s. Depth increases about 8 m over this period. The small effect of compression on the results can be eliminated by commenting out compression data on *Root*.**geo** or adding the following to *Root*.**sim** within the scope of the simulation:

```
Settings
   NoCompress
```

This introduces **Settings** input which is used to change general parameter settings in a free or MLD simulation. For example, free runs use Nørsett's adaptive timestep Runge-Kutta integration [1,8,9] by default. We can change parameters such as the minimum and maximum timesteps, and the tolerance used to determine a successful timestep:

```
Settings
   TimeStep   0    5.0     ! increase max only
   Tolerance  0.0025       ! reduce tolerance
```

For a simple straight-ahead run, these settings almost triple the speed of simulation (see the real/CPU time ratio on *Root*.**spr**) because most of the timesteps are at the new maximum. But if an autopilot is active there is little change because few timesteps get to reach either maximum value. Altering tolerance has only a modest effect.

The Adams-Bashforth method of integration is generally, subject to the chosen timestep, somewhat slower than Nørsett for simple simulations, but faster if one or more autopilots are running. It is invoked by :

```
AdamsBash
```

within the scope of **Settings**. Calculation and output time steps are optional arguments; by default they are 0.1 s and 1 s respectively. Because the calculation timestep

is fixed, this method is potentially less accurate than Nørsett's, although there is often little difference in the results.

To demonstrate the effect of setting a differential blow, create the following short simulation:

```
Free
   Ukt      4
   Depth   65
   Settings
      AdamsBash
    0      AutoDepth 110
    0      Prop       82
  107.9    Flood   1   0.08   ! from a previous sim !
  140.     Blow  All
  300      Stop
```

Unlike the previous simulation that started with a flood during a depth change, this attempt at recovery consists of simply blowing both MBTs. It will be found that the pitch angle gets steadily more negative and the simulation is stopped by zero forward speed before the surface is reached. This constitutes a failed recovery. Add the following within the scope of **Settings** to delay blowing the aft MBT until a pitch up angle of five degrees is reached:

```
DiffBlow   5
```

Run the program. Apart from a brief transient, the pitch angle remains positive and recovery is successful.

Supplementary Exercises

- Try the other initialization inputs; note how long the perturbations affect the dynamics of the submarine. The impact of most is relatively shortlived.
- Try the other **Settings** options. See how those that modify calculations in the program, such as **Isothermal** and **LumpNoNpr**, affect the results.
- Run some more realistic operational manoeuvers and again look for the sensitivity of the results to relevant program parameters.

Exercise 11. MLD Curve Calculations

Initialize and **Settings** inputs were described in the previous exercise. The only differences for a MLD simulation are that **SpeedStop** and **LSpeedStop** inputs are fully-functioning, and that the default integration method is Adams-Bashforth. The latter can be altered with **Settings** → **NorsettRK**.

Figure A.13 shows a schematic MLD and associated nomenclature. It comprises a series of safe operating envelopes bounded on the left by a flood avoidance curve (or zero speed) and on the right by sternplane jam curves, each for an allowable control deflection δ_i.

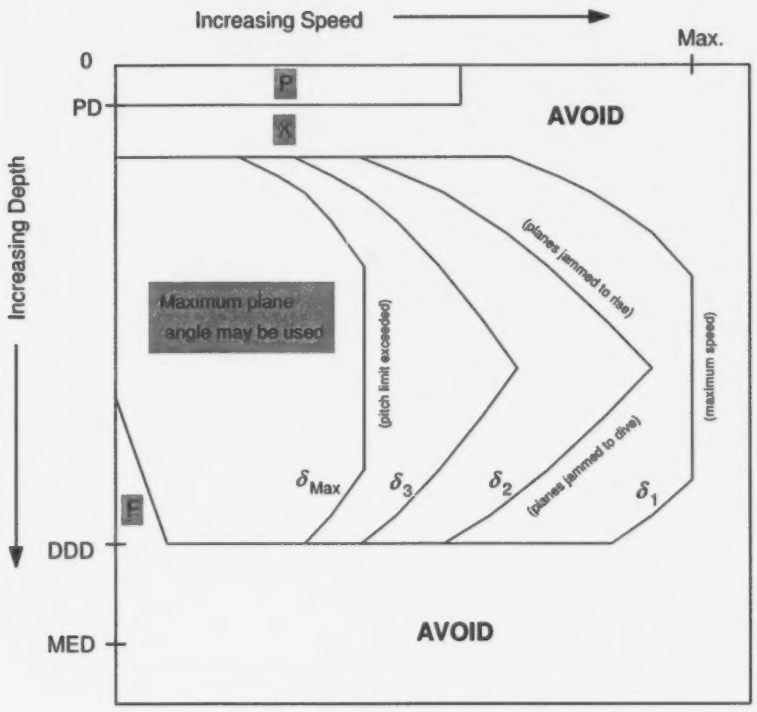


Figure A.13 Schematic Manoeuvering Limitation Diagram.

Speed:	Max.	maximum submerged speed
Depth:	PD	periscope depth
	DDD	deep diving depth
	MED	maximum excursion depth
Operating Regions:	F	flood avoid zone (FAZ)
	P	transition from periscope depth to surface only
	X	avoid where deep draft surface vessels operate
Parameters:	$\delta_i,\ i = 1, 2, 3, \text{Max}$	magnitude of plane deflection, $\delta_1 < \delta_2 < \delta_3 < \delta_{\text{Max}}$

This is the conventional form of a MLD; unorthodox control arrangements may lead to others. The MLD simulation option in `DSSP50` is used to calculate the left and right boundaries; parameters such as DDD and MED are predetermined.

The core requirements of MLD input are that it provide a range of depth and a range of speed for the calculations, and that the simulation type — flood or plane jam — be identified. We will start by creating a basic MLD flood simulation. The following can be added to an existing *Root*`.sim` file or put on a new one:

```
MLD
  Depth
     295   290  275
     250   200  150
     100    50
  Ukt       0  20  0.05
  Excursion   0   300
  Stop         300
  Flood
    Breach  #FwdComp  0.15  !  hole location and diameter
```

This defines eight depths and a range of 0 to 20 kt over which to calculate the flood boundary. The `Excursion` record defines the depth excursion limits; for a flood these should correspond to the surface and to the MED or charted depth. `Stop` defines the time limit for each run in the simulation. Runs implicitly start at 0 s. The number of runs at each depth is n in $(2^{n-1} < U_{max}/\epsilon_U \leq 2^n)$, where maximum speed U_{max} and resolution ϵ_U are set by `Ukt` (or by a `Ums` record if in m/s). Here, $U_{max} = 20$ and $\epsilon_U = 0.05$, so $n = 9$.

Run the program. Progress is indicated by a summary line for each run on the monitor. The MLD curve data are output on a `sim`*nnn*`.txt` file and included on *Root*`.spr`. Run summaries are output on a `aux`*nnn*`.txt` file by default.

The curve is plotted on figure A.14. The small avoidance area at shallow depth results from exceeding the default pitch limit of 30 degrees during recovery.

If more than a summary of each run in the simulation is required, adding the following record will put time histories of the state variables and other significant parameters on the `aux`*nnn*`.txt` file:

```
AuxOutput  Verbose
```

The resulting file is quite large. Individual runs can be copied from it for plotting or other analysis. Run numbers are combined with the label of the first column in each table: `0001Ts`, `0002Ts`, etc. It is recommended that the user keep verbose `aux`*nnn*`.txt` output set while doing the rest of this exercise so that the effect of various input options can be seen more clearly.

There are a couple of features of this example that make the flood avoidance region somewhat larger than typical. First, the hole is fairly large for an unsecured flood, and

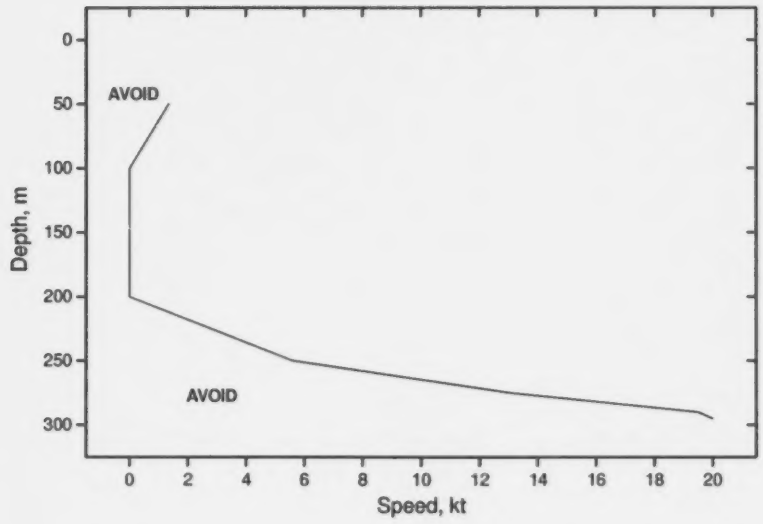


Figure A.14. Example flood avoidance boundary.

second, the depth range extends nearly to the maximum excursion. The latter condition is appropriate for a shallow water flood curve for which the flood avoidance zone is often called the bottom avoidance zone. For a deep water flood calculation, the depth range is normally to the DDD.

Pitch and roll limits can be changed. There are also factors of safety that can be applied to calculated depth, pitch, and roll that allow for uncertainty in the calculations. Thus, for a pitch limit of 30 degrees and a corresponding factor of 1.2, the limit is considered to be exceeded at a calculated pitch of $30/1.2 = 25$ degrees. For non-zero pitch, depth can be calculated at aft and forward locations on the submarine before comparing with a limit [1]. Set values for these options as follows (unless stated otherwise, additional input is within the scope of the `MLD` record):

```
PitchLimit    35
RollLimit     35
Factor   1.1   1.1   1.1  ! depth, pitch, roll
PitchCorrD  -25   25     ! distance aft,fwd of ref for depth(pitch)
```

This modest relaxation of the limits noticeably reduces the flood avoidance zone. Other parameters that generally have a significant effect include flood detection and time delays.

Flood detection is triggered by the increase in mass (in practice, the volume of water that is indicated by float switches in the bilge), by the resulting depth excursion, by the resulting pitch excursion, or after a time delay. The first of these to occur starts the

recovery sequence. The responsible trigger will be flagged on *Root*`.log` if the following record is placed at the top of *Root*`.sim`:

```
Output Verbose
```

However, this option produces a large quantity of other data and extraneous warnings, and it may be easier to identify the trigger directly from time histories on `aux`*nnn*`.txt`. With the current defaults, flood recovery is generally triggered by the increase in mass. This can be altered by setting the mass criterion to be very large or by specifying only depth and pitch triggers by putting the following within the scope of `Flood`:

```
Trigger    dp
```

The resulting flood curve is very much more restrictive. Note that flood detection can be made instantaneous by adding a 't' or 'T' to the argument of the record above and specifying a time criterion of zero. The result is much less restrictive. Trigger criteria are specified with `MassCrit`, `DepthCrit`, `PitchCrit`, and `TimeCrit` records.

Time delays are applied to various actions following flood detection: MBT blowing, plane deflection, increasing RPM, and securing the flood if that action is selected. For example, add the following within the scope of `Flood`:

```
DelayBlow  10
DelaySecur 20  !  (ignored unless Secured is present)
Secured
```

The flood curve is even less restrictive, this time because the flood is secured after 20 s.

Supplementary Exercises

- Set the maximum excursion to 355 m. Given the MLD depth range, this is more appropriate for a deep water flood. The previous flood curve should be relaxed at points where before it was limited by depth exceedence.
- Try the other options available. These include `ManRise` (use a modest plane angle), `RecDepth`, `RecRPM`, and `StopBlow` within the scope of `Flood`, and various `Initialize` and `Settings` options.
- Try different hole sizes in different locations. See the effect of losing power for a flood in the aft WTC. In that case, it might be appropriate to also increase some of the time delays.
- Do for a different version of the submarine to compare, for example, the recovery of a X-rudder configuration with that of a +-rudder configuration (there may not be much difference).

A sternplane jam simulation has jam input in place of flood input with other MLD, initialization, and settings input essentially as before. Note that existing flood input does not have to be removed or commented out; it is sufficient to add an `Ignore` record. (This is similarly used to ignore jam input.) Nevertheless, for the present exercise we will start with a new basic jam simulation.

First note that at the start of the simulation the jammed planes (identified by JamSpec) move from their initial position to their jammed position. There is a delay before the jam is recognized followed by additional delays for each of the recovery actions. The standard recovery procedure is to slow to a recovery speed at which the jammed planes have insufficient control authority to constitute a problem, while using unjammed planes to counteract the effect of the jam.

Put the following on *Root*.sim:

```
MLD
  Depth
     300   290  275            ! Set range DDD to PM, or vice versa,
     250   200  150            ! for a jam
     100    50   25
      20
  Ukt       0  20  0.05
  Excursion  15   355          !  PD to MED
  Stop         300
  AuxOutput  Verbose
  Jam
    JamSpec  SternPlane  -10   !  Jam to rise
    RecRPM       -150          !  Recovery - astern RPM
```

Ordering astern RPM is the only recovery action in this example. Run the program. Speed is limited over most of the depth range to about 12.6 kt for the component based model or about 13.3 kt for the coefficient based model. In what follows, we will assume the former. Numerical results will differ depending on what version of the submarine is used.

To aid recovery with bowplanes or sailplanes, add the following record within the scope of Jam:

```
ManJRec  ForePlane   -20   !  planes to dive
```

As a result the jam boundary is relaxed considerably: speed over most of the depth range is limited to about 16 kt. The two jam boundaries are compared on figure A.15.

This is by default a permanent jam. Make it temporary by defining a delay, following detection, after which the planes unjam. However, they will not move unless given a recovery angle (e.g., by another ManJRec record). Add the following within the scope of Jam:

```
ManJRec  SternPlane   20  !  reverse jam angle
DelayUnJ   15
```

It will be found that the resulting jam boundary is more restrictive, not less as we might expect. The reason is that reversing the jam angle on the sternplanes results in the pitch angle being exceeded before the recovery speed, by default zero, is achieved. Try different sternplane angles for recovery. Using zero should give a less restrictive boundary.

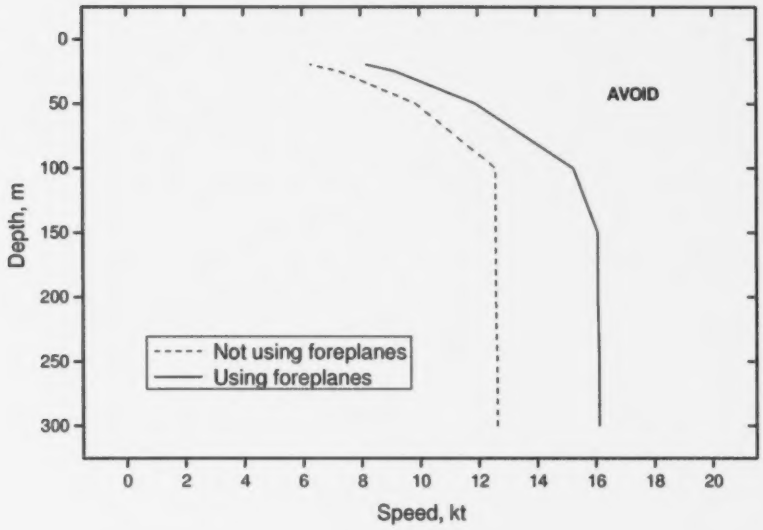


Figure A.15. Example avoidance boundaries for a 10 degree jam to rise, with and without using foreplanes for recovery.

A still better strategy for recovery may be to use the depth autopilot to try to maintain initial depth. Try it by replacing the `ManJRec` (Manual Jam Recovery) records with a single `AutoJRec` record. In some respects, using the autopilot for recovery may better represent the actions of a helmsman than using the 'manual' option, but this depends on the response prescribed by the official recovery procedures.

A plane jam simulation can be given initial conditions. It is common to associate an initial angle of pitch with a jam that makes the submarine pitch in the same sense, thus representing a worst case scenario. In `DSSP40` this was handled so that a single curve for both jam to rise and jam to dive could be generated for a given jam angle, applying the appropriate initial pitch in each case, but with `DSSP50` the jam to rise and jam to dive curves have to be generated separately. Doing so facilitates producing the MLD in slide-rule form, in which the jam to dive curve appears on the inner slide and the jam to rise curve is on a transparent outer jacket. Moving the slide relative to the jacket approximates the effect on the MLD of changing the depth of the bottom (i.e., the depth limit and bottom avoidance zone). Furthermore, the rise and dive curves are virtually symmetrical unless blowing is used for recovery, so generally the data of one can simply be reflected into the other.

We can add the alternative condition for recovery that the depth excursion simply be reversed within depth limits before the recovery speed is reached. This is done with the `Transient` record [1] within the scope of `Jam`. In the run summary, it is typical that

successful intermediate runs terminate with '**Depth Excursion OK**' instead of '**Boat Stopped OK**', but equally typical that there is little or no effect on the jam curve.

A dive stop is a hardware or software limit on control deflections to dive that is activated when speed exceeds a given value. In **DSSP50** it is modeled by the more general concept of applying different control deflection limits above and below a given initial speed. This applies only to a MLD simulation and is specified within the scope of **Initialize** in place of **DeltaLim** for an individual appendage:

```
Initialize
  SpeedStop  #PortSternPlane          25  -25
             8.   Ukt                 25  -10  !  10 deg limit to dive
  SpeedStop  #StarboardSternPlane     25  -25
             8.   Ukt                 10  -25  !   "  "    "    "   "
```

or in place of **LDeltaLim** for a lump control mode:

```
Initialize
  Sternplane  1
    LSpeedStop   25    -25
         8 Ukt   10    -25
```

If the requested jam exceeds whichever limit corresponds to the initial speed, the jam is set to the limit without any warning. An initial deflection that exceeds the limit is similarly set to the limit without warning.

Supplementary Exercises

- Simulate a jam to dive curve corresponding to one of the jam to rise curves produced above, and confirm that they are symmetrical.
- Add blowing to recovery with **Blow**, **DelayBlow**, and **StopBlow** records [1]; there should be some asymmetry between corresponding jam to rise and jam to dive curves.
- Observe the effect of making pitch and other initial parameters nonzero. Try the effect of various **Settings** parameters.
- See the difference, probably small, due to how the submarine is modeled; e.g., compare results for the component based and equivalent coefficient based versions of the +-rudder configuration.

Finally, it is suggested that the user exercise the program by generating the MLD curves for a realistic set of operational requirements.

Annex B. Program Unit Lists

Build 090910 source code is organized in a series of Fortran files as follows. The program unit names are cited in error messages.

`Dssp50.for`; the highest level routines:

Unit	Function
`DSSP50`	main program
`GEO_L1`	process *Root*`.geo` level 1 input
`GEO_L2_AUTO`	process *Root*`.geo` autopilot level 2 input
`GEO_L2_CMP`	process *Root*`.geo` `Compress` level 2 input
`GEO_L2_HPA`	process *Root*`.geo` `HPAir` level 2 input
`GEO_L2_HUL`	process *Root*`.geo` `Hull` level 2 input
`GEO_L2_LFT`	process *Root*`.geo` `Lift` level 2 input
`GEO_L2_LMP`	process *Root*`.geo` `Lump` level 2 input (inactive)
`GEO_L2_MBT`	process *Root*`.geo` `MBT` level 2 input
`GEO_L2_OOP`	process *Root*`.geo` `OOPCalc` level 2 input
`GEO_L2_PLT`	process *Root*`.geo` `Plot` level 2 input
`GEO_L2_PRP`	process *Root*`.geo` `Propulsor` level 2 input
`GEO_L2_TCP`	process *Root*`.geo` `Trim`, `Comp`, and `TrimComp` level 2 input
`GEO_L2_WIN`	process *Root*`.geo` `WagBInit` level 2 input
`GEO_L2_WTC`	process *Root*`.geo` `WTC` level 2 input
`INP_L2_REF`	process *Root*`.geo` and *Root*`.sim` `Reference` level 2 input
`READ_MYP`	process `DSSP50.myp` input
`SIM_L1`	process *Root*`.sim` level 1 input
`SIM_L2_CAP`	process *Root*`.sim` `Captive` level 2 input
`SIM_L2_COF`	process *Root*`.sim` `Coefficnt` level 2 input
`SIM_L2_FRE`	process *Root*`.sim` `Free` level 2 input
`SIM_L2_MLD`	process *Root*`.sim` `MLD` level 2 input
`SIM_L2_STA`	process *Root*`.sim` `Static` level 2 input

`Sp50cint.for`; time simulation integration:

Unit	Function
`ADBASH`	first order Adams-Bashforth predictor
`DDNORK`	order (3,4) Runge-Kutta integrator
`INTPC0`	fourth order interpolation, C^0 continuous
`INTPC1`	fourth order interpolation, C^1 continuous

`Sp50esam.for`; added mass estimation:

This file comprises the core subroutine `ESAMCR` and associated program units adapted, with only minor modifications, from the `ESAM` added mass program [10]. To avoid future naming conflicts the program units are listed here: `ESAMCR`, `AMMATX`, `AMOUTP`, `BCZB`, `CHGSGN`, `CHIR`, `COS2Z`, `INERCO`, `QKE`, `SWITCH`, `UEU`, `VFIELD`, and `ZEROIF`.

`Sp50prep.for`; geometry and systems preprocessing:

Unit	Function
`ALBON`	hull axial potential flow using Albone's methods
`ALBONC`	hull axial potential flow for a hull with a closed tail
`ALBONO`	hull axial potential flow for a hull with an open tail
`AXIBBL`	hull axial flow thin boundary layer
`D_ASCOF`	set up lump coefficients
`DELLIM`	process DeltaLim and SpeedStop records
`DL3_COEF`	process lump coefficient input
`HEOOPX`	axis rotation matrix for OOP calculation
`HULLAX`	check hull local axes specification
`HULLGE`	process hull geometry
`HULLHY_HUL4`	hull hydrodynamic calculations using the `HulFor` method
`HULLH1_HUL4`	hull linear damping from `HulFor`
`HULLPT`	save hull geometry definition for *Root*`.gpl`
`JNP_L3_LMP`	process lump control parameters
`JST_L4_LMP`	process initial lump control parameters
`LIFTDL`	appendage orientation parameters
`LIFTGE`	process appendage geometry
`LIFTHY`	appendage hydrodynamic calculations
`LIFTLO`	appendage local axis calculations
`LIFTPT`	save appendage geometry definition for *Root*`.gpl`
`LUMPIO`	transfer lump data to or from a file
`LUMPPT`	save lump visualization for *Root*`.gpl`
`MLD_L3_FLD`	process MLD flood recovery data
`MLD_L3_JAM`	process MLD jam recovery data
`PLOTGE`	manipulate geometry for plotting
`PLOTXA`	write *Root*`.gpl` in `AcroSpin` format
`PLOTXM`	write *Root*`.gpl` in `MatLab` format
`PLOTXP`	write *Root*`.gpl` in `Maple` format
`PLOTXV`	write *Root*`.gpl` in VRML 1 format
`PLOTXW`	write *Root*`.gpl` in VRML 2 format (inactive)

RUDCOF	Bohlmann-Spreiter method for tailplane efficiency
SIM_L3_INI	process free or MLD initialization
SIM_L3_SET	process free or MLD settings
SPREIT_WB	Spreiter's loading function for K_{WB}
SPREIT_W_B	Spreiter's loading function for k_{WB}
THWAIT	hull axial laminar boundary layer — Thwaite's method
TRUCK	hull axial turbulent boundary layer — Truckenbrodt's method
WAGBINIT	initialize Wageningen B propellers
XK1	evaluate Albone's virtual mass function

`Sp50sims.for`; simulation run processing:

Unit	Function
AUTOCOM	set up parameters for autopilot commands
AUTODOIT	update autopilot error and control signals
AUTOFIND	process autopilot command records
BLOCOM	interpret `Blow` commands
BLOWVENT	update BG and MBT data during a ballast blow or vent
COEF_D	hydrodynamic coefficient calculation with the DERIVS method
COEF_L	hydrodynamic coefficient calculation with the LLSR method
COMAND	process free simulation command records
D_ABCD	three/four point coefficient evaluation for COEF_D
DELCOM	process `Delta` commands
DELFIND	interpret `Delta` command strings
DYDT	define the extended equations of motion, $\dot{\mathbf{u}} = F(\mathbf{u}, t)$
FLOOD	update WTC data during a flood
FLUDCOM	interpret `Flood` commands
HEADFIND	process heading argument strings
MLDOUT	output MLD run time histories
PASSFAIL	evaluate MLD pass-fail conditions
PRPCOM	process `Propulsor` command records
SPCOUT	output free simulation to `sim`*nnn*`.txt` and `aux`*nnn*`.txt`
SPRINT	execute a free simulation sprint
STABILITY	small perturbation stability analysis
TCMPCOM	process `Trim` and `Comp` commands
TIMHXN	estimate loads for a captive time history
TRIM_COMP	update trim and compensation status
VENTCOM	interpret `Vent` commands

`Sp50subs.for`; general subroutines:

Unit	Function
ADDMGEN	generate component added masses
ADDMMAT	setup added mass matrix in reference axes
CANINC	calculate floodwater volume and centroid
CANMOX	function for floodwater longitudinal moment
CANMOZ	function for floodwater vertical moment
CANVOL	function for floodwater volume
CFTURB	turbulent flow friction drag coefficient
CFWET	estimate hull friction drag
DELASSOC	set up control appendage `Delta` command associations
DOCOMPR	estimate compression
DOPRVC	estimate control gains for plane reversal
DOSTAT	sum static loan estimates
DYNAMV	estimate dynamic loads for the complete vehicle
FLUDCG	estimate floodwater centroid as a function of volume and pitch
HULL_LOOK	in-plane hull load lookup tables
HYDRST	rationalize **BG**, **CB**, and **CG**
INRTMAT	setup inertial matrix in reference axes
IWEIGHTS	process `WeightDCI` records
KTKQ02	`DSSPtwo` propulsor model
KTKQST	`STRprop` propulsor model
KTKQWB	`WageningB` propulsor model
NLOCGO	save local axis systems in vehicle axes
NREFGO	adjust local axis systems to reference axes
OOPGRADE	update out-of-plane time histories
OOPSET	static out-of-plane calculation
OOPSTEP	dynamic out-of-plane calculation
PRPFRC	calculate propulsor loads
PRPINT	initialize propulsion condition
PRPINX	estimate initial propulsor loads
RATCOM	rationalize simulation commands and conflicts
REVPPT	process `CB`, `CG`, or `Reference` redefinition
STATHL	`Hull` static forces and moments
STATHL_HUL4	`HulFor` static load calculation
STATLM	`Lump` static forces and moments (inactive)
STATLT	`Lift` static forces and moments

`Sp50util.for`; low level utilities:

Unit	Function
ADDRNG	add vector to a double precision ring array
CHCOMP	word string comparison
CIRCIH	returns angle in range $-\pi$ to π
CIRCIT	returns angle in range 0 to 2π
COMP3F	compare number triplets
CROOTS	obtain roots of cubic equation
DECOMP	matrix decomposition by Gaussian elimination and partial pivoting
DEFLBL	initialize the component label array
DEPMES	handler for deprecated procedures
GETCHR	get the next word in the input stream
GETDBL	convert the next word in the input stream to double precision
GETINT	convert the next word in the input stream to integer
GTDRNG	get vector from a double precision ring array
HELP	write program information to *Root*`.log`
INDICT	find dictionary keyword matching a character string
INDRNG	insert vector into a double precision ring array
INLABL	find component label matching a character string
INTRPL	interpolation of a single valued tabulated function
LLSR	linear least squares regression
LOOKUP2	lookup table bilinear interpolation
MESSAG	error message manager
NEARNESS	distance from a point to a line
OPENIT	open a file with status checks
PCQUAD	quadrature using piecewise cubics
PYTHAG	robust solution of $\sqrt{a^2 + b^2}$
QSFM	fourth-order quadrature, for evenly-spaced abscissae
SOLVE	solve linear system of equations decomposed by DECOMP
SVBKSB	solve linear system of equations decomposed by SVDCMP
SVDCMP	matrix singular value decomposition (SVD method)
TIQUAD	return coefficients for a three-point quadratic fit
TOLABL	add an entry to the component label list
UCCHAR	character case conversion and filtering
WORDFLAG	parse a word for flag characters
WORDIN	find a word within a character string
YD_CHEK	range check for double precision

In addition, the following 'Include' files are required to define common blocks and shared parameters:

```
Sp50aaaa.cmn
Sp50adel.cmn
Sp50aioa.cmn
Sp50aplt.cmn
Sp50auto.cmn
Sp50axbl.cmn
Sp50coef.cmn
Sp50ggeo.cmn
Sp50labl.cmn
Sp50mldg.cmn
Sp50stat.cmn
Sp50stuf.cmn
```

Annex C. Error Messages

Error messages are issued to both the terminal and *Root*`.log`. They have four levels of severity: fatal, warning with pause, warning, and information. Fatal error messages are generated by a condition or problem that forces termination of program execution. A warning with pause indicates an actual or potential problem for which the user may wish to terminate program execution (the default response is to continue). Simple warnings are issued to indicate an actual or potential problem while allowing program execution to continue automatically. In a few instances, an information message is issued; this indicates that the program is flagging, at worst, a benign anomaly that the user should be aware of.

The message format is:

```
    !! Fatal Error in program_unit_name
or  ?? Warning from program_unit_name
or  -- Information from program_unit_name
       message line 1
       message line 2
```

If a warning includes a pause, it is followed by:

```
    ?? Continue execution ?  ...  [Y]/N
```

and the user's response is noted in the *Root*`.log` file as one of:

```
    ?! Execution terminated by user
or  ?? Execution continued by user
```

Message line 2 is to clarify the error report on line 1, and is only present if required.

As the program evolved, some error conditions became trapped in higher-level routines and corresponding lower level error checks are now redundant. Such errors, if reported, are tagged on line 2 as '`- impossible`'.

Other error messages may include the notation '`- system error`' on line 2, indicating a problem reported by the compiler or operating system. These are typically '`ERR=`' exits from a Fortran `READ`, `WRITE`, or `OPEN` command. The last, for example, may result from an attempt to open an output file that is already open in another program.

Syntax and other errors in user input should be straightforward to identify because the input is echoed record-by-record to the *Root*`.log` file, and in most cases the error message will be displayed just after the offending input record.

An example of each type of error message follows:

```
!! Fatal Error in GEO_L2_HUL
   Unsavable label
   - label string is duplicated or invalid

?? Warning from GEO_L1
   Autopilot depth error gain (CZZ)
   - could not be initialized (no hull)
?? Continue execution ?  ...  [Y]/N

?? Warning from GEO_L2_LMP
   Lump CG defaults to Lump Location

-- Information from GEO_L2_HUL
   HULFOR: number of stations not defined
   - default (21) assumed
```

A handful of fatal errors have a solution that is not immediately obvious. Some of them are summarized here, others are noted elsewhere in the documentation.

A truncated or very bluff hull afterbody may result in the following:

```
!! Fatal Error in HULLH1_HUL4
   No minimum found in dA/dx (DT2) for separation
   - very bluff afterbody?  - force separation
```

indicating that the hull load algorithm has failed to find a minimum needed in the slope of the afterbody cross sectional area to locate flow separation. The user should check for irregularities in the afterbody offset data. If they appear satisfactory, the region of likely separation, if known, can be specified with a `HullSep` record.

Time integration with Nørsett's method may generate an error of the form:

```
!! Fatal Error in SPRINT
   Unexpected IND = N from DDNORK (interpolated/stepped output)
   - inconsistent timesteps and/or tolerance ??
```

where N is one of

- `-1`: an internal limit on the number of function evaluations has been exceeded; this condition should not be possible in `DSSP50`.
- `-2`: the current minimum timestep is greater than the maximum; this should not be possible in `DSSP50` unless these quantities have been input incorrectly with a `TimeStep` record.
- `-3`: the integrator is unable to satisfy the accuracy required to continue. Possible causes of this are that integration tolerance is too small relative to the minimum timestep or that the matrix of inertial plus added mass properties is ill-conditioned. The latter can alternatively result in an unexpected stop with the message '`DYDT: sim`

`stopped by zero/negative forward speed`'. The key here is that the stop is unexpected: the program stops with this message because u ceases to be positive (as would legitimately occur in simulating a crash-back, for example), but the prior time history is inconsistent with that happening. If the problem is associated with a reference shift, it may be avoided by redefining the vehicle's target point instead of the reference.

A captive model simulation may fail with the error '`TIMEHISTRY file P data is inconsistent`' (or '...`Q`...', or '...`R`...'). This generally indicates a fundamental error in the user-supplied state variable time histories.

Note that a fatal error message will not make it to the log file if the file is closed with the message still spooled for output. (The message will appear momentarily in the command window, but that will close also.) This typically happens for multiple simulations with the error occuring towards the end of the *Root*`.sim` file. Making the questionable simulation first in the sequence will often overcome the difficulty.

Although it is intended that any problem results in an error message from the program, there may be undiscovered conditions that halt the program with an error message from the compiler library. An example (contrived by introducing a bug into the code!) from the standard Windows executable release is shown in figure C.1. If input triggering the message can be identified, it may be possible to work around the problem until the bug is properly fixed.

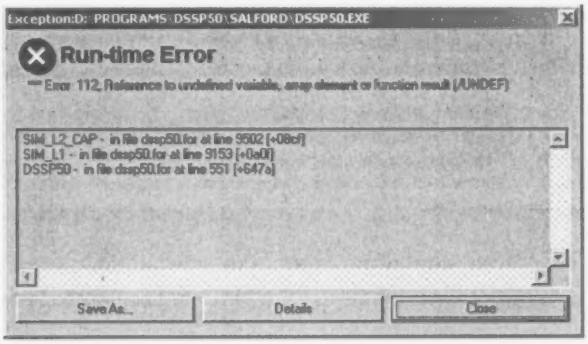

Exception:D:\PROGRAMS\DSSP50\SALFORD\DSSP50.EXE
Run-time Error
Error 112, Reference to undefined variable, array element or function result (UNDEF)
SIM_L2_CAP - in file dssp50.for at line 9502 [+08cf]
SIM_L1 - in file dssp50.for at line 9153 [+0a0f]
DSSP50 - in file dssp50.for at line 551 [+647a]
Save As...
Details
Close

Figure C.1. Run time error message from the Salford compiler library.

If any of the following are encountered, please report them to DRDC Atlantic together with the program build number, associated input and output files, and any other relevant information: errors tagged '`- impossible`', time integrator errors with $N =$ `-1` (and $N =$ `-2` if `TimeStep` appears correct), and compiler library errors.

This page intentionally left blank.

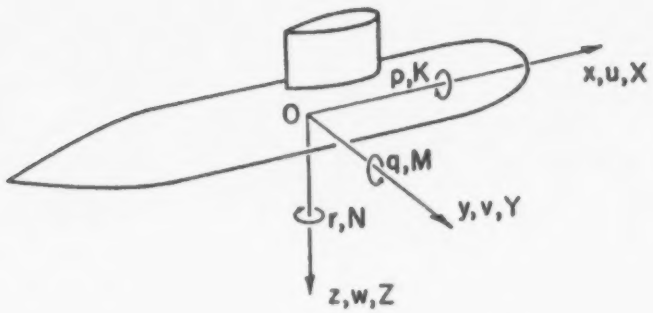
List of symbols/abbreviations/acronyms/initialisms

Symbols

b_{exp}	exposed appendage span
b_1, b_2	spanwise location of inboard and outboard flap extent
B, B_i	hull breadth, of station i
$\overline{BG}$	metacentric height
$\overline{c}$	mean appendage chord
$\overline{c_f}$	mean flap chord
C_{Do}	lump pressure drag
d_m	lump diameter
D	maximum hull diameter
k	out-of-plane load distribution parameter
K, M, N	roll, pitch, and yaw moment
ℓ	reference length for nondimensionalizing, typically LBP
ℓ_m	lump length
L	hull length
p, q, r	roll, pitch, and yaw rate
r_M	maximum hull radius
t	time
T, T_i	hull height, of station i
u, v, w	axial, lateral, and normal velocity
U	total velocity
U_{max}	MLD maximum speed
x, y, z	vehicle axis coordinates or local axes
x_e, y_e, z_e	chart axes
X, Y, Z	axial, lateral, and normal forces
α	angle of attack
β	angle of drift
δ	control surface deflection
δ_i	i^{th} sternplane deflection limit on a MLD
ϵ_U	MLD speed tolerance

θ	pitch angle
ρ	water density
ϕ	roll angle, or appendage anhedral or dihedral angle
ψ	yaw angle, heading

Reference Axes and Loads



Acronyms and Abbreviations

CB	Center of Buoyancy
CF	Canadian Forces
CG	Center of Gravity
DDD	Deep Diving Depth
HP	High Pressure
LBP	Length Between Perpendiculars
LCB	Longitudinal CB
MBT	Main Ballast Tank
MDTF	Marine Dynamic Test Facility
MED	Maximum Excursion Depth
MLD	Manoeuvering Limitation Diagram
PD	Periscope Depth
PMM	Planar Motion Mechanism
VRML	Virtual Reality Modeling Language
WTC	WaterTight Compartment

Distribution

DRDC Atlantic TM 2009–226

LIST PART 1: Internal Distribution by Center:

2 Author (1 hard copy, 1 CD)

3 DRDC Atlantic Library (1 hard copy, 2 CDs)

5 TOTAL LIST PART 1

LIST PART 2: External Distribution within Canada by DRDKIM

1 Library and Archives Canada

1 NDHQ/DRDKIM 3

1 NDHQ/DMSS 2

1 NDHQ/DMEPM (SM) 4–2

1 CFMWC – Attn: N. Sponagle
PO Box 99000 STN Forces, Halifax NS, B3K 5X5

1 Micromedia Limited, Acquisitions/Research Indexes
20 Victoria St., Toronto ON, M5C 2N8

1 NRC Institute for Ocean Technology – Attn: Director General
PO Box 12093 Station A, St. John's NL, A1B 3T5

7 TOTAL LIST PART 2

LIST PART 3: External Distribution outside Canada by DRDKIM

1 Defence Research Information Center
Station Square House, St. Mary Cray, Orpington Kent, England BR5 3R3

1 DPA
Ash 3c #3311, MOD Abbey Wood, Bristol, England BS34 9JH

1 MoD – Attn: Director of Naval Architecture and Marine Engineering
PO Box 20702, 2500 ES The Hague, The Netherlands

1 MARIN – Attn: Head of Naval Research
PO Box 28, 6700 AA Wageningen, The Netherlands

1 Navy Office – Attn: Director Naval Ship Design
Department of Defence CP1–505, Canberra ACT 2600, Australia

1 Document Exchange Center, Defence Information Services Branch
Department of Defence CP2–5–08, Canberra ACT 2600, Australia

1 Bundesamt für Wehrtechnik und Beschaffung – SG 13
Postfach 7360, 6057 Koblenz, Germany

---

7 TOTAL LIST PART 3

LIST PART 4: External Distribution outside Canada by DRDC Atlantic

3 Under covering letter from Canadian Project Officer, IEP ABCA–02–01:

NSWCCD – Attn: David Walden, Project Officer ABCA–02–01
9500 MacArthur Blvd, Bethesda MD, USA 20817–5700

DSTL Winfrith – Attn: Simon Tinker, Project Officer ABCA–02–01
Bldg A32 Winfrith Technology Center, Dorchester Dorset, England DT2 8WX

DNPD – Attn: Martin Grimm, Project Officer ABCA–02–01
Department of Defence CP4–5–074, Canberra ACT 2600, Australia

---

3 TOTAL LIST PART 4

---

22 **TOTAL COPIES REQUIRED**

DOCUMENT CONTROL DATA		
1. ORIGINATOR Defence R&D Canada – Atlantic	2. SECURITY CLASSIFICATION Unclassified	
3. TITLE DSSP50 Build 090910 Documentation, Part 1: Guide and Tutorial		
4. AUTHORS M. Mackay		
5. DATE OF PUBLICATION December 2009	6a. NO. OF PAGES 78	6b. NO. OF REFS 10
7. DESCRIPTIVE NOTES DRDC Atlantic Technical Memorandum		
8. SPONSORING ACTIVITY		
9a. PROJECT OR GRANT NO. 11GX01	9b. CONTRACT NO.	
10a. ORIGINATOR'S DOCUMENT NUMBER DRDC Atlantic TM 2009–226	10b. OTHER DOCUMENT NOS.	
11. DOCUMENT AVAILABILITY Unlimited		
12. DOCUMENT ANNOUNCEMENT (if different from 11)		
13. ABSTRACT DSSP50 is a computer code for evaluating the dynamics and manoeuverability of submarines and smaller streamlined underwater vehicles. This document is the guide to the program. It comprises an overview of DSSP50 and a tutorial. Other documents detail the input requirements and describe the underlying algorithms.		
14. KEYWORDS Underwater Vehicles Submarines Manoeuvering Simulation		

This page intentionally left blank.

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE R&D DÉFENSE

www.drdc-rddc.gc.ca